UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSGRADO

SECCIÓN EN CIENCIAS

Modelos y métodos de optimización para problemas de scheduling difusos en máquinas simples

Para optar por el grado académico de Magíster en Matemática Aplicada

Autor: Lic. Edwin Raúl Lazo Eche

Asesor: Mg. Flabio Alfonso Gutierrez Segura

Febrero, 2017

PIURA - PERÚ

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSGRADO

SECCIÓN EN CIENCIAS

PROGRAMA DE MAESTRÍA EN MATEMÁTICA APLICADA

MODELOS Y MÉTODOS DE OPTIMIZACIÓN PARA PROBLEMAS DE SCHEDULING DIFUSOS EN MÁQUINAS SIMPLES

APROBADA EN CONTENIDO Y ESTILO POR

MG. EDGARD JHONY OJEDA MAURIOLA

PRESIDENTE

MG. JUAN PANTA COBEÑAS

SECRETARIO

MG. AMERICO CARRASCO TINEO

VOCAL

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSTGRADO

SECCIÓN EN CIENCIAS

PROGRAMA DE MAESTRÍA EN MATEMÁTICA APLICADA

TESIS

MODELOS Y MÉTODOS DE OPTIMIZACIÓN PARA PROBLEMAS DE SCHEDULING DIFUSOS EN MÁQUINAS SIMPLES

LOS SUSCRITOS DECLARAN QUE EL PRESENTE TRABAJO ES ORIGINAL EN SU CONTENIDO Y FORMA

LIC. EDWIN RAÚL LAZO ECHE

EJECUTOR

MG. FLABIO GUTIERREZ SEGURA

ASESOR

ESCUELA DE POSGRADO

UNIVERSIDAD NACIONAL DE PIURA

ACTA DE SUSTENTACIÓN

MAESTRÍA EN MATEMÁTICA APLICADA

Los Miembros del Jurado Calificador que suscriben, reunidos para la sustentación de la Tesis, para optar el Grado Académico de Magister en **MATEMÁTICA APLICADA.** Presentada por:

LAZO ECHE – EDWIN RAÚL

Con el asesoramiento del M.Sc. FLABIO ALFONSO GUTIERREZ SEGURA, denominada:

"MODELOS Y MÉTODOS DE OPTIMIZACIÓN PARA PROBLEMAS DE SCHEDULING DIFUSOS EN MÁQUINAS SIMPLES"

Oídas las respuestas y absueltas las observaciones formuladas, se declara:

APROBADO				DESAPROBADO
Excelente	*Sobresaliente*	*Bueno*	*Aceptable*	
_____	_____	X	_____	_____

En consecuencia, previa aprobación del Art.º 83, del Reglamento General de la Escuela de Posgrado, queda en condiciones de ser calificado **APTO** para obtener el Grado Académico de **MAGISTER EN MATEMÁTICA APLICADA.** De conformidad con lo estipulado en la ley.

PIURA, VIERNES 17 DE MARZO DEL 2017.

M.Sc. EDGAR JOHNY OJEDA MAURIOLA
PRESIDENTE

M.Sc. JUAN PANTA COBEÑAS
SECRETARIO

M.Sc. MAERICO CARRASCO TINEO
VOCAL

Resumen

Los problemas de scheduling en máquinas se definen como una asignación factible entre un conjunto de trabajos en una determinada planificación, las cuales pueden desarrollarse bajo diferentes condiciones tanto para las máquinas como para el entorno de trabajo, con el objetivo de optimizar una determinada función objetivo.

En este trabajo se desarrollan modelos y métodos para problemas de scheduling en máquinas simples, considerando la presencia de restricciones de precedencia así como la ausencia del mismo, en donde la información correspondiente al conocimiento del estado de la duración de las tareas es imprecisa. Los métodos y modelos obtenidos han sido implementados en software de naturaleza simbólica así como, software especializado en optimización y satisfacción de restricciones. En el procesamiento de los métodos y modelos se ha utilizado la aritmética difusa como herramienta de programación de tareas y obtención de schedule factibles para los problemas planteados. Los modelos han sido aplicados en benchmarks referentes a fuzzy scheduling siendo óptimo en el cálculo del máximo tiempo de culminación ponderada.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Scheduling en máquinas

Es común hablar en la industria manufacturera de la planificación, entendida como un elemento importante para el posicionamiento y éxito de la misma, en un mercado que asume como premisa fundamental la competitividad. Las empresas no solamente buscan incursionar y posicionarse en el mercado, sino obtener esa competitividad eficiente que garantice de manera directa la satisfacción de los receptores de los productos elaborados.

Muchos de los parámetros que intervienen en la planificación dependen de los recursos tecnológicos, entre ellos máquinas, velocidades, tiempos de procesamiento, además de la capacidad de los operarios para procesar las órdenes de entrega de productos, las fechas en que vencen las diversas entregas, los costos de mantenimiento preventivo de las máquinas y en fin, de acuerdo al tipo de industria que se desarrolle aparecerán diversos factores necesarios para la planificación, que a la vez hacen que esta operación constituya un problema serio de toma de decisiones. Como es de observarse, la toma de decisiones para una

eficiente planificación no siempre resulta ser tarea sencilla, para ello existen un conjunto de técnicas y formas de obtener secuencias de trabajos para procesar un conjunto de tareas en unas máquinas, este proceso en la literatura se le denomina scheduling, específicamente scheduling en máquinas.

Las empresas planifican con miras a cumplir optimizar ciertas funciones objetivos, por citar algunos: minimizar el tiempo de culminación de las tareas planificadas, minimizar tardanzas en la planificación, minimizar la asignación de valores óptimos para las funciones de costo, entre otras. Una correcta asignación de tareas a máquinas puede permitir manejar de manera eficiente la capacidad tecnológica de las mismas, en otras palabras, evitar que algunas máquinas estén encendidas en tiempos innecesarios lo cual afecta directamente en su tiempo de vida y pueda originar interrupciones en la planificación realizada.

En este capítulo se estudia la teoría de scheduling desde el punto de vista clásico, tanto en sus formulaciones, supuestos y variables que intervienen.

1.1. Scheduling

Scheduling o la programación de tareas, es un término de nuestro vocabulario muy presente, aunque a menudo no tengamos una definición concreta. En realidad, aunque no hablamos de programación o scheduling, sí es familiar hablar de horarios, y es que un horario es un producto de una programación o un scheduling,

al que también se denomina schedule. Un schedule es un planificación tangible (evidenciable) como lo es un horario de clases, el horario de partida en una estación de autobuses, otras.

Intuitivamente podemos pensar en scheduling como el proceso de elaborar un schedule, aunque raras veces caigamos en cuenta cómo se realiza este proceso. De hecho, aunque pensamos en un programa como algo tangible, el proceso de programación parece bastante intangible, hasta que lo consideramos con cierta profundidad. A menudo se aborda el problema en dos pasos: secuenciación y programación. En el primer paso, se planea una secuencia o se decide cómo seleccionar la siguiente tarea. En el segundo paso, planeamos la hora de inicio, y tal vez el tiempo de finalización, de cada tarea.

En el proceso de programación, es necesario saber el tipo y la cantidad de cada recurso para que se pueda determinar cuándo las tareas pueden llevarse a cabo de manera factible. Cuando se especifican los recursos, se define efectivamente el límite del problema de scheduling. Además, describimos cada tarea en términos de información como su requisito de recursos, su duración, el tiempo más temprano en el que puede comenzar y el momento en que se debe completar. En general, la duración de la tarea es incierta, pero es posible que desee suprimir esa incertidumbre al declarar el problema. También debemos describir las limitaciones tecnológicas (restricciones de precedencia) que existen entre las tareas. La información sobre recursos y tareas define un problema de programación. Sin embargo, encontrar una solución es a menudo una cuestión

bastante compleja, y los enfoques formales de solución de problemas son útiles.

Para la programación es necesario recurrir a modelos de ayuda para entender el problema de scheduling y encontrar una buena solución, uno de estos modelos es el diagrama de Gantt (Ver figura 1.1). En su forma básica, el gráfico de Gantt muestra la asignación de recursos en el tiempo, con recursos específicos mostrados a lo largo del eje vertical y una escala de tiempo mostrada a lo largo del eje horizontal. El gráfico básico de Gantt supone que los tiempos de procesamiento se conocen con certeza, pero esto no siempre ocurre y tal fenómeno es motivo de este trabajo.

Con un diagrama de Gantt, podemos descubrir información sobre un schedule dado analizando las relaciones geométricas. Además, podemos reorganizar tareas en el gráfico para obtener información comparativa sobre horarios alternativos. De esta manera, el diagrama de Gantt sirve como una ayuda para medir el rendimiento y comparar horarios, así como para visualizar el problema en primer lugar.

El diagrama de Gantt puede mostrar en el eje vertical otras representaciones tales como, las prioridades o ponderaciones de un trabajo (en el caso de máquinas simples o single machine).

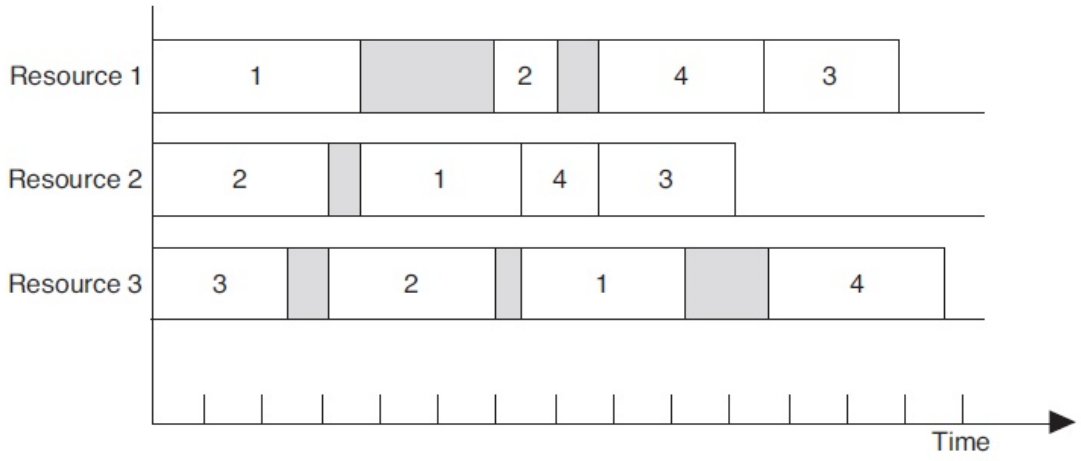

Figura 1.1: Diagrama de Gantt Clásico (Tomado de [7])

1.1.1. Parámetros y Supuestos clásicos en scheduling

En todo problema de scheduling se considera el número de trabajos y las máquinas, ambas cantidades finitas. El número de máquinas es denotado de m y por n al número de trabajos. Usualmente, el subíndice j se refiere al trabajo, mientras el subíndice i se refiere a la máquina. Si un trabajo requiere una etapa de procesamiento u operaciones, entonces el par (i, j) refiere a la etapa de procesamiento u operación del trabajo j en la máquina i.

A menudo en un problema de scheduling se consideran parámetros que contienen información especial sobre las características de los trabajos, así se tiene:

- **Tiempo de procesamiento.** Denotado por p_{ij}, representa el tiempo de procesamiento del trabajo j en la máquina i. El último índice puede omitirse en caso se trabaje con una sola máquina.

- **Periodo de lanzamiento.** Se denota por r_j y es el momento en que el trabajo llega al sistema, es decir, la hora más temprana en la que el trabajo j puede iniciar su procesamiento.

- **Fecha de vencimiento.** Se denota por d_j y representa la fecha de envío o de finalización comprometida (es decir, la fecha en que se promete el trabajo al cliente).La finalización de un trabajo después de su fecha de vencimiento está permitida, pero entonces se incurre en una penalidad.

- **Peso o ponderación.** Denotada por ω_j y es básicamente un factor de prioridad, denotando la importancia del trabajo j relativo a los otros trabajos en el sistema. Por ejemplo, este peso puede representar el costo real de mantener el trabajo en el sistema. Este costo podría ser un costo de mantenimiento o inventario; también podría representar la cantidad de valor ya añadido al trabajo.

En la naturaleza clásica de scheduling existen diversos supuestos que definen los entornos y cómo se establecerá la solución de los mismos. De entre éstos podemos destacar.

- El conjunto de trabajos y máquinas se conocen y son fijos.

- Todos los trabajos y máquinas están disponibles en el mismo tiempo y son independientes.

- Todas las máquinas permanecen disponibles durante un período ilimitado de tiempo.
- El tiempo de procesamiento por cada trabajo en todas las máquinas es fijo, se puede conocer su distribución de probabilidad.
- Los tiempos de preparación están incluidas en los tiempos de procesamiento.
- Un tamaño de lote básico se fija para todos los trabajos.
- Todos los trabajos tienen la misma prioridad.
- No se permite el derecho preferente.
- A cada trabajo se le asigna una fecha de vencimiento.
- Cada trabajo es procesado por todas las máquinas asignadas.
- Cada máquina procesa todos los trabajos que le son asignados.
- Se conoce un plan de procesamiento de trabajos y es fijo.

1.1.2. Categorías en scheduling

Las diferentes categorías en las que se ubican los problemas de scheduling provienen de las relajaciones de ninguna, una o varias restricciones, las cuales lo pueden categorizar en:

- **Determinístico.** Cuando los elementos del problema, tales como el estado de llegada de los trabajos, tiempos de procesamiento, fechas de vencimiento, ordenamiento, disponibilidad de máquinas no incluyen factores estocásticos y se conocen a priori.

- **Estático.** Es el mismo problema determinístico excepto que la nturaleza de las llegadas de los trabajos es diferente. El conjunto de trabajos no cambia y están disponibles de antemano.
- **Dinámico.** A diferencia del estático, el conjunto de trabajos cambia y las llegadas ocurren en diferentes momentos.
- **Estocático.** Uno de los factores que intervienen presentan naturaleza estocástica.

1.1.3. Entornos de scheduling

Graham en [26], presenta una notación para describir los entornos de scheduling basándose en 3 elementos principales.

$$\alpha \mid \beta \mid \theta$$

donde

α : describe la notación del entorno de máquina (Ver tabla 1.1).

β : explica las características del procesamiento y restricciones (Ver tabla 1.2).

θ : contiene la información sobre la función objetivo (Ver tabla 1.3).

De acuerdo con Pinedo [52], describiremos los posibles entornos para las máquinas.

Tabla 1.1: Notaciones comunes para Entorno de máquina α

Nombre del entorno	Notación	Descripción
Máquina simple	1	Una máquina
Máquinas paralelas idénticas	P_m	m : número de máquinas
Máquinas paralelas con diferente rapidez	Q_m	m : número de máquinas
Flow shop	F_m	m : número de máquinas
Job shop	J_m	m : número de máquinas
Open shop	O_m	m : número de máquinas

- **Single Machine.** O máquinas simples, denotada por 1, es uno de los casos más sencillos de los posibles entornos, sin embargo pueden convertirse en casos especiales de otros más complicados.

- **Máquinas paralelas idénticas.** Denotada por P_m, donde m es la cantidad de máquinas en paralelo. El trabajo j requiere una sola operación y puede ser procesado en cualquiera de las m máquinas o en cualquiera que pertenezca a un subconjunto dado. Si el trabajo j no puede ser procesado en cualquier máquina, pero sólo en cualquier perteneciente a un subconjunto específico M_j, entonces la entrada M_j aparece en el campo β.

- **Máquinas paralelas con diferente rapidez.** Denotada por Q_m, donde m es la cantidad de máquinas en paralelo con diferente rapidez. La rapidez de la

máquina i es denotada por v_i. El tiempo p_{ij} empleado por el trabajo j en la máquina i es calculado mediante $\frac{p_j}{v_i}$ (suponiendo que el trabajo j recibe todo su procesamiento desde la máquina i).

- **Flow Shop.** Denotado por F_m, donde m es el número de máquinas en serie. Cada trabajo debe ser procesado en cada una de las máquinas. Todos los trabajos tienen que seguir la misma ruta, es decir, tienen que ser procesados primero en la máquina 1, luego en la máquina 2, y así sucesivamente. Una vez completada una máquina, un trabajo se une a la cola en la siguiente máquina. Por lo general, se supone que todas las colas operan bajo la disciplina FIFO (First In First Out), es decir, un trabajo no puede ̈pasarä otro mientras espera en una cola. Si la disciplina FIFO está en efecto, el Flow Shop se denomina Flow Shop con permutación y el campo β incluye la entrada prmu que se detallará adelante.

- **Job Shop.** Denotado por J_m, en una Job Shop con m máquinas cada trabajo tiene su propia ruta predeterminada a seguir. Se hace una distinción entre las Jb Shop en las que cada trabajo visita cada máquina a lo sumo una vez y los Job Shop en las que un trabajo puede visitar cada máquina más de una vez. En este último caso, el campo β contiene la entrada rcrc para la recirculación.

- **Open Shop.** Denotado por O_m, y m es el número de máquinas. Cada trabajo debe ser procesado de nuevo en cada una de las máquinas. Sin

embargo, algunos de estos tiempos de procesamiento pueden ser cero. No hay restricciones con respecto al enrutamiento de cada trabajo a través del entorno de la máquina. El planificador puede determinar una ruta para cada trabajo y diferentes trabajos pueden tener rutas diferentes.

Tabla 1.2: Notaciones comunes para las características del procesamiento β

Término	Notación	Descripción
Período de lanzamiento	r_j	Una máquina no puede iniciar su procesamiento antes de un tiempo r_j
Preferencia	Prmp	Un trabajo puede ser interrumpido con la llegada de otro con mayor prioridad
Restricción de precedencia	Prec	Cuando el inicio de un trabajo depende de la finalización de otro.
Averías	Brkdwn	Las máquinas no están continuamente disponibles
Recirculación	Recrc	Cuando un trabajo visita una máquina más de una vez
Permutación	Prmu	Las órdenes de procesamiento de todos los trabajos en una máquina se mantiene en el taller.

Las características de las restricciones de procesamiento pueden hallarse en:

- **Período de lanzamiento.** Si este símbolo aparece en el campo β, entonces el trabajo j no puede iniciar su procesamiento antes de su fecha de lanzamiento r_j. Si r_j no aparece en el campo β, el procesamiento del trabajo j puede comenzar en cualquier momento. En contraste con las fechas de lanzamiento, las fechas de vencimiento no se especifican en este campo. El tipo de función objetivo proporciona una indicación suficiente de si hay o no fechas de vencimiento.

- **Derecho preferente.** Implica que no es necesario mantener un trabajo en una máquina, una vez iniciado, hasta su finalización. El programador puede interrumpir el procesamiento de un trabajo (preempt) en cualquier momento y poner un trabajo diferente en la máquina. La cantidad de procesamiento que se ha recibido ya no se pierde. Cuando un trabajo es interrumpido se vuelve a colocar después en la máquina (o en otra máquina en el caso de máquinas paralelas), sólo necesita la máquina para su tiempo de procesamiento restante. Cuando se aceptan derechos preferentes el símbolo p̈rmpsë incluye en el campo β; cuando prmp no está incluido, no se permiten las preferencias.

- **Restricciones de precedencia.** Las restricciones de precedencia pueden aparecer en una sola máquina o en un entorno de máquinas paralelas, lo que requiere que se deba completar uno o más trabajos antes de que se permita a otro trabajo iniciar su procesamiento. Existen varias formas

especiales de restricciones de precedencia: si cada trabajo tiene como máximo un predecesor y como máximo un sucesor, las restricciones se denominan cadenas. Si cada trabajo tiene como máximo un sucesor, las restricciones se denominan intree. Si cada trabajo tiene como máximo un predecesor, las restricciones se denominan outtree. Si no aparece prec en el campo β, los trabajos no están sujetos a restricciones de precedencia.

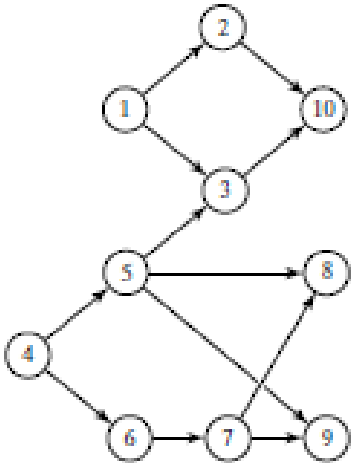

Figura 1.2: Ejemplo de un grafo de precedencias

- **Procesamiento en lote.** Denotado por $batch(b)$, en el cual una máquina puede ser capaz de procesar una serie de trabajos, digamos b, simultáneamente; es decir, puede procesar un lote de hasta b trabajos al mismo tiempo. Los tiempos de procesamiento de los trabajos en un lote pueden no ser todos iguales y el lote completo sólo se termina cuando se ha completado el último trabajo del lote, lo que implica que el tiempo de finalización de todo el lote viene determinado por el trabajo con el más largo Tiempo de procesamiento. Si b = 1, entonces el problema se reduce a un entorno de programación con-

vencional. Otro caso especial que es de interés es $b = \infty$, es decir, no hay límite en el número de trabajos que la máquina puede manejar en cualquier momento.

- **Averías.** Denotado por $brkdwn$, donde las averías de la máquina implican que una máquina puede no estar disponible continuamente. Los períodos en los que no se dispone de una máquina se consideran fijos en esta parte del libro (por ejemplo, debido a cambios o mantenimiento programado). Si hay un número de máquinas idénticas en paralelo, el número de máquinas disponibles en cualquier punto en el tiempo es una función del tiempo, es decir, $m(t)$. A veces, las averías de la máquina también se denominan limitaciones de disponibilidad de la máquina.

- **Permutaciones.** Una restricción que puede aparecer en el entorno de Flow Shop es que las colas delante de cada máquina funcionan de acuerdo con la disciplina FIFO (First In First Out). Esto implica que el orden (o permutación) en el cual los trabajos pasan por la primera máquina se mantiene en todo el sistema.

Detallaremos algunas funciones objetivo las que por lo general son minimizados.

- **Makespan**. El makespan (C_{mx}) es definido por

$$C_{mx} = \max(C_1, C_2, ..., C_n)$$

y es equivalente al tiempo de culminación del último trabajo a ser procesado. Un makespan mínimo usualmente indica una buena utilización de la(s)

Tabla 1.3: Notaciones comunes para las funciones objetivo θ

Término	Notación
Makespan	C_{mx}
Máximo retraso	L_{mx}
Tiempo de culminación ponderada	$\sum \omega_j C_j$
Tardanza total ponderada	$\sum \omega_j T_j$
Tiempo de culminación total	$\sum C_j$

máquina(s).

- **Máximo retraso**. El máximo retraso(L_{mx}) es definido como

$$L_{mx} = \max(L_1, L_2, ..., L_n)$$

donde el retraso por cada trabajo es definido como

$$L_j = C_j - d_j$$

la que es positivo cuando el trabajo j se completa tarde y negativo cuando se completa temprano. Esta medida es entendida también como la mayor incidencia de violación a las fechas de vencimiento.

- **Tiempo de culminación ponderada**. Se formula como $\sum \omega_j C_j$. La suma de los tiempos de culminación ponderados de los n trabajos proporciona una indicación de los costes totales de tenencia o de inventario incurridos por el schedule.

- **Tardanza total ponderada**. Se formula como $\sum \omega_j T_j$ donde la tardanza del trabajo j está dada por

$$T_j = \max\left(C_j - d_j; 0\right) = \max\left(L_j; 0\right).$$

Esta es también una función de coste más general que el tiempo de culminación total ponderado.

La diferencia entre la tardanza y el retraso reside en el hecho de que la tardanza nunca es negativa. La pena unitaria del trabajo j se define como

$$U_j = \begin{cases} 1 & \text{si } C_j > d_j \\ 0 & \text{Otro caso} \end{cases} \tag{1.1}$$

Todas las funciones objetivo arriba mencionadas son medidas de desempeño regulares. Una medida de rendimiento regular es una función no decreciente en $C_1, C_2, \cdots, C_n$. Sin embargo existen otras medidas de rendimiento no regulares. Por ejemplo, cuando un trabajo j tiene una fecha de vencimiento d_j puede estar sujeto a una penalidad por la anticipación de la entrega (earliness), donde esta anticipación para el trabajo j es definido por:

$$E_j = \max\left(d_j - C_j; 0\right).$$

Esta penalidad por anticipación de entrega es una función no creciente en $C_1, C_2, \cdots, C_n$.

1.2. Procedimientos generales propuestos para scheduling

Pinedo en [52], describe una serie de procedimientos de propósito general que son útiles para tratar problemas de programación en la práctica y que pueden implementarse con relativa facilidad en los sistemas de programación industrial. Todas las técnicas descritas son heurísticas que no garantizan una solución óptima; Sino que buscan encontrar soluciones razonablemente buenas en un tiempo relativamente corto. Las heurísticas tienden a ser bastante genéricas y pueden adaptarse fácilmente a una gran variedad de problemas de scheduling.

1.2.1. Dispatching Rules

La investigación en Dispatching Rules ha estado activa durante varias décadas y muchas reglas diferentes han sido estudiadas en la literatura. Estas reglas pueden ser clasificadas de varias maneras. Por ejemplo, se puede hacer una distinción entre reglas estáticas y dinámicas. Las reglas estáticas no dependen del tiempo. Son sólo una función del trabajo y/o de los datos de la máquina, por ejemplo, WSPT. Las reglas dinámicas dependen del tiempo. Un ejemplo de una regla dinámica es la regla del mínimo Slack (MS) que ordena trabajos según $\max(d_j - p_j - t, 0)$ que depende del tiempo. Esto implica que en algún momento el trabajo j puede tener una prioridad más alta que el trabajo k y en algún momento posterior los trabajos j y k pueden tener la misma prioridad.

Las siguientes son algunas de las reglas de despacho (dispatching rules) que

han sido investigadas desarrolladas e implementadas en varias de las investigaciones en scheduling y que son referentes para este trabajo.

SPT o SEPT

Por su siglas en inglés, SPT (Tiempo de procesamiento más corto) o SEPT (tiempo de procesamiento esperado más corto) tienen en común procesar primero los trabajos con menor tiempo de procesamiento en máquina. Para este criterio existen varias versiones.

- **SRPT:** Más corto tiempo de procesamiento restante.
- **TSPT:** Menor tiempo de procesamiento truncado. El trabajo con más pequeño tiempo de procesamiento se procesa primero, pero si hay un trabajo con un tiempo de espera de operación mayor que W, ese trabajo se procesa primero, W se elige arbitrariamente.
- **WSPT:** Más corto tiempo de procesamiento promedio. El trabajo con la más pequeña proporción se procesa primero. la proporción se calcula dividiendo el tiempo de procesamiento de la operación con su respectivo peso.

EDD

Earliest Due Date (por sus siglas en inglés EDD). El trabajo con la fecha de vencimiento más próxima se procesa primero. Existen algunas variantes para esta regla

- **ODD:** Fecha de vencimiento de la operación. La operación con más pequeña fecha de vencimiento se procesa primero.
- **MDD:** Fecha de vencimiento modificada. Desde el conjunto de trabajos en espera de una máquina específica, los trabajos se asignan una nueva fecha de vencimiento, y el EDD es medido en ese conjunto.

CR

En la proporción crítica (Critical ratio), el trabajo con la más pequeña proporción se procesa primero. El CR se determina dividiendo la asignación de trabajo por el tiempo de trabajo restante

1.2.2. Métodos de búsqueda local

La siguiente sección considera algoritmos del tipo de mejora. Los algoritmos del tipo de mejora son conceptualmente completamente diferentes de los algoritmos del tipo constructivo. Empiezan con un programa completo, que puede ser seleccionado arbitrariamente, y luego tratar de obtener un mejor horario mediante la manipulación de la programación actual. Una clase importante de algoritmos de tipo de mejora son los procedimientos de búsqueda local. Un procedimiento de búsqueda local no garantiza una solución óptima. Por lo general, intenta encontrar un schedule que es mejor que el actual en la vecindad del actual. Dos schedules son vecinos, si uno puede ser obtenido a través de una modificación bien definida de la otra. En cada iteración, un procedimiento de búsqueda local realiza una búsqueda dentro del vecindario y evalúa las diversas soluciones vecinas. El procedimiento acepta o rechaza una solución candidata como el siguiente calendario

para moverse, basado en un criterio de aceptación / rechazo dado.

Se pueden comparar los distintos procedimientos de búsqueda local con respecto a los siguientes cuatro criterios de diseño:

- La representación de la programación necesaria para el procedimiento.
- El diseño del vecindario.
- El proceso de búsqueda dentro del vecindario.
- El criterio de aceptación-rechazo.

La representación de un schedule puede ser a veces no trivial. Un schedule sin derecho preferente en máquinas simples se puede especificar mediante una simple permutación de los n trabajos. Un schedule en un taller de trabajo sin derecho preferente puede especificarse por m secuencias consecutivas, cada una representando una permutación de n operaciones en una máquina específica. Basándose en esta información, se pueden calcular los tiempos de inicio y de finalización de todas las operaciones. Sin embargo, cuando se permiten las preferencias, el formato de la representación del schedule se vuelve significativamente más complicado.

El diseño de la vecindad es un aspecto muy importante de un procedimiento de búsqueda local. Para una sola máquina, una vecindad de un schedule particular puede ser simplemente definida como todas las programaciones que se pueden obtener haciendo un solo intercambio de parejas adyacente. Esto implica que hay $n - 1$ schedules en la vecindad de la programación original. Una vecindad

más grande de un schedule de una sola máquina puede ser definida tomando un trabajo arbitrario en el horario e insertándolo en otra posición en el horario. Claramente, cada trabajo se puede insertar en $n - 1$ posiciones. El vecindario entero contiene menos de $n(n - 1)$ vecinos ya que algunos de estos vecinos son idénticos. La vecindad de un schedule en un ambiente de máquina más complicado es generalmente más compleja.

1.2.3. Algoritmos Genéticos

Los algoritmos genéticos son más generales y abstractos que la búsqueda tabú. La búsqueda de tabú puede, de cierta manera, ser vistos como casos especiales de algoritmos genéticos. Los algoritmos genéticos, cuando se aplican al scheduling, se visualizan los schedules o programas como individuos o miembros de una población. Cada individuo se caracteriza por su aptitud (fitness). La aptitud de un individuo se mide por el valor asociado de la función objetivo. El procedimiento funciona iterativamente, y cada iteración se conoce como una generación. La población de una generación consta de sobrevivientes de la generación anterior más los nuevos horarios, es decir, los descendientes (hijos) de la generación anterior. El tamaño de la población generalmente permanece constante de una generación a la siguiente. La descendencia se genera a través de la reproducción y mutación de los individuos que formaban parte de la generación anterior (los padres). Los individuos a veces también se conocen como cromosomas.

En un entorno máquinas múltiples, un cromosoma puede consistir en sub-

cromosomas, cada uno de los cuales contiene la información relativa a la secuencia de trabajo en una máquina. Una mutación en un cromosoma progenitor puede ser equivalente a un intercambio pairwise adyacente en la secuencia correspondiente. En cada generación los individuos más aptos se reproducen mientras mueren los menos aptos. Los procesos de nacimiento, muerte y reproducción que determinan la composición de la siguiente generación pueden ser complejos y, por lo general, dependen de los niveles de aptitud de los individuos en la generación actual.

Un algoritmo genético, como un proceso de búsqueda, difiere en un aspecto importante de la búsqueda tabu. En cada paso iterativo se generan una serie de schedules diferentes y se transfieren al siguiente paso. En la búsqueda tabú sólo se transporta un solo schedule de una iteración a la siguiente. Por lo tanto, la búsqueda tabú puede considerarse como caso especial de algoritmos genéticos con un tamaño de población igual a 1. Este esquema de diversificación es una característica importante de los algoritmos genéticos. En los algoritmos genéticos, el concepto de vecindario tampoco se basa en un único calendario, sino en múltiples horarios. El diseño del vecindario de la población actual de schedules se basa, por lo tanto, en técnicas más generales que las utilizadas en la búsqueda de tabú. Un nuevo schedule se puede generar combinando partes de diferentes horarios de la población actual. Un mecanismo que crea un nuevo horario se conoce a menudo como un operador de crossover.

1.2.4. Optimización por colonias de hormigas

Los algoritmos de optimización de colonia de hormigas (ACO) combinan técnicas de búsqueda local, reglas de distribución y otras técnicas dentro de un marco. El paradigma ACO está inspirado en el rastro que sigue el comportamiento de las colonias de hormigas. Las hormigas, al moverse a lo largo de un camino a un destino, dejan a lo largo de su trayectoria un producto químico llamado feromona como señal para que otras hormigas sigan. Un algoritmo de ACO asume que una colonia de hormigas (artificiales) construye iterativamente soluciones para el problema en cuestión usando rastros de feromonas (artificiales) que están relacionados con soluciones previamente encontradas, así como con información heurística. Las hormigas comunican entre sí sólo indirectamente a través de cambios en las cantidades de feromonas que depositan en sus senderos durante la ejecución del algoritmo. Debido a que las soluciones construidas por las hormigas pueden no ser óptimas localmente, muchos algoritmos ACO permiten a las hormigas mejorar sus soluciones a través de un procedimiento de búsqueda local.

1.3. Máquinas Simples

El problema de máquinas simples ilustra una variedad de temas de scheduling en un modelo manejable. Proporciona un contexto en el que se puede investigar muchas medidas de rendimiento diferentes y varias técnicas de solución. Por lo tanto, es un elemento fundamental en el desarrollo de una comprensión global de los conceptos de programación.

Para entender completamente el comportamiento de un sistema complejo, es vital comprender sus partes, y muy a menudo el problema de máquinas simples aparece como parte de un problema de scheduling más grande. A veces, incluso puede ser posible resolver el problema incrustado de una sola máquina independientemente y luego incorporar el resultado en el problema más grande. Por ejemplo, en los procesos de múltiples operaciones, puede existir una etapa de cuello de botella y el tratamiento del cuello de botella por sí mismo con análisis de una sola máquina puede determinar las propiedades de toda la programación. En otras ocasiones, el nivel en el que las decisiones deben ser tomadas puede dictar que los recursos deben ser tratados en conjunto, como si los puestos de trabajo vinieran a una sola instalación [7].

1.3.1. Problema $1||\sum \omega_j C_j$

El objetivo de este problema es el tiempo de culminación ponderada $1||\sum \omega_j C_j$. El peso w_j del trabajo j puede considerarse un factor de importancia; puede representar un coste de tenencia por unidad de tiempo o el valor ya añadido al trabajo j. Este problema da lugar a una de las reglas más conocidas en la teoría de programación, la llamada regla de tiempo de procesamiento más corto ponderado (WSPT). Los tiempos de procesamiento de cada trabajo se denota por p_j, lo cual a su vez permite definir los tiempos de culminación de cada trabajo, denotados por C_j.

El objetivo de este tipo de problemas consiste en encontrar un schedule que minimice $\sum \omega_j C_j$. Un enfoque clásico que se tiene para $\sum \omega_j C_j$ consiste en considerar ω_j una tasa de costo de almacenamiento de un recurso necesario para procesar un determinado trabajo J_j.

Los primeros aportes para la solución de este problema, datan desde 1960, con la implementación de la *Regla de Smith*. Esta metodología clásica, proporciona una idea basada en el ordenamiento a través de la razón entre los tiempos de procesamiento y sus respectivos pesos; con esto, es posible encontrar una secuencia factible no decreciente (Weighted Shortest Processing Time o simplemente WSPT) que minimice este máximo tiempo de culminación promedio.

Para elaborar el modelo se hace teniendo en cuenta como únicas variables $C_j, j = 1, ..., n$, donde cada C_j representa el tiempo de culminación de cada trabajo en un schedule dado,[15] y sus respectivos pesos o ponderaciones ω_j. Así el objetivo (desde el punto de vista de la programación lineal) consiste en:

$$\min \sum_{j=1}^{n} \omega_j C_j \tag{1.2}$$

Un lema importante que interviene en la formulación es el que sigue:

Lema 1 *Sea C una solución factible para la formulación del tiempo de culminación y sin pérdida de generalidad sea $C_1 \leq C_2 \ldots \leq C_n$. Entonces, se cumple que :*

$$C_j \geq \frac{1}{2} \sum_{k=1}^{j} p_k \tag{1.3}$$

□

Los ambientes en que se desarrolla este problema dependen mucho del enfoque en que se quisiera evaluar y solucionar el modelo.

El criterio de la proporción de Smith brinda una herramienta para resolver el problema $1||\sum \omega_j C_j$ en orden $O(n \log n)$, ya que es capaz de establecer una secuencia ordenada,simplemente teniendo en cuenta las proporciones $\frac{\omega_j}{p_j}$ asociadas a cada trabajo J_j.

Veamos el siguiente teorema que establece una secuencia óptima para este tipo de problemas:

Teorema 1 : *Una secuencia es óptima para* $1||\sum \omega_j C_j$ *sí y solo sí los trabajos están ordenados de forma no decreciente por sus proporciones* $\frac{p_j}{\omega_j}$.

□

En efecto, supongamos que existe un trabajo J_k que está precedido inmediatamente por J_j con

$$\frac{p_j}{\omega_j} > \frac{p_k}{\omega_k}.$$

Si el tiempo de culminación de J_k es C_k entonces J_j se completa en tiempo $C_k - p_k$.

El efecto que se logra al intercambiar estos trabajos es para obtener una secuencia con costos positivos, en efecto:

$$\begin{aligned} [\omega_j (C_k - p_k) + \omega_k C_k] - [\omega_k (C_k - p_j) + \omega_j C_k] &= \omega_k p_j - \omega_j p_k \\ &= \omega_j \omega_k \left(\frac{p_j}{\omega_j} - \frac{p_k}{\omega_k} \right) > 0. \end{aligned}$$

□

Dicho de este modo, el proceso consiste en minimizar de un único criterio de optimalidad que es no decreciente en cada uno de los tiempos de culminación.

Corolario 1 *Una secuencia es óptima para* $1||\sum C_j$ *si y solo si los trabajos se ordenan de manera creciente con respecto a sus tiempos de procesamiento* p_j

□

Si queremos dar una idea más intuitiva, supongamos que tenemos n trabajos ejecutados en el orden $1, 2, \ldots, n$. Entonces

$$\begin{aligned} C_1 &= p_1 \\ C_2 &= p_1 + p_2 \\ C_3 &= p_1 + p_2 + p_3 \\ &\vdots \\ C_j &= p_1 + p_2 + \cdots + p_j \end{aligned}$$

efectuando una suma simple, el makespan, viene dado por:

$$\sum_{i=1}^{j} C_j = (j)\, p_1 + (j-1)\, p_2 + (j-2)\, p_3 + \cdots + 2p_{j-1} + p_j. \tag{1.4}$$

Observando 1.4, esto significa que se debe hacer una asignación de los coeficientes $\{1, 2, \ldots, \mathrm{j}\}$

En el siguiente ejemplo se muestra la utilización de esta secuencia y su comparación con otra secuencia obtenida de manera aleatoria.

Ejemplo **1.1** Consideremos el siguiente problema de máquina simple.

Trabajo	1	2	3	4	5	6
p_j	8	12	7	16	9	8
ω_j	4	10	4	3	8	10

Determinar una secuencia de trabajos que minimice el máximo tiempo de culminación ponderada.

Solución:

Usaremos la secuencia SPT para generar un schedule optimo, para ello calculemos las proporciones de cada uno de los trabajos.

Trabajo	p_j	ω_j	$\frac{p_j}{\omega_j}$
1	8	4	2
2	12	10	1.2
3	7	4	1.75
4	16	3	5.3
5	9	8	1.125
6	8	10	0.8

Ordenando de acuerdo con las proporciones obtenidas

Trabajo	p_j	ω_j	$\frac{p_j}{\omega_j}$	C_j	$\omega_j C_j$
6	8	10	0.8	8	80
5	9	8	1.125	17	136
2	12	10	1.2	29	290
3	7	4	1.75	36	144
1	8	4	2	44	352
4	16	3	5.3	60	180

Entonces, la secuencia dada por el schedule $S = \{6, 5, 2, 3, 1, 4\}$, minimiza (en comparación con otras secuencias) el tiempo de culminación promedio a $1182\ u$, es decir

$$\sum_{j=1}^{6} \omega_j C_j = 1182 \tag{1.5}$$

Para someter a tal prueba, generemos otra secuencia esta vez escogida de manera aleatoria $S^* = \{4, 2, 3, 6, 1, 5\}$

Trabajo	p_j	ω_j	C_j	$\omega_j C_j$
4	3	3	3	9
2	12	10	15	150
3	7	4	22	88
6	8	10	40	400
1	8	4	48	192
5	9	8	57	456

Según estos resultados, para el schedule S^* el tiempo de culminación ponderada es:

$$\sum_{j=1}^{6} \omega_j C_j = 1295$$

que tal y como puede compararse es mayor al obtenido en 1.5.

El siguiente paso es dar una interpretración geométrica de estos resultados y su posible visualización.

Intuitivamente, para tener una idea geométrica de lo que ocurre, podemos utilizar la representación bi-dimensional Gantt, colocando como eje horizontal p, en cuya representación se encuentran los tiempos de procesamiento p_j con sus respectivas unidades, y el eje vertical ω_j en cual se ubican los respectivos pesos.

Si aplicamos la *Regla de Smith* gráficamente se obtendría el arreglo mostrado en la figura 1.3.

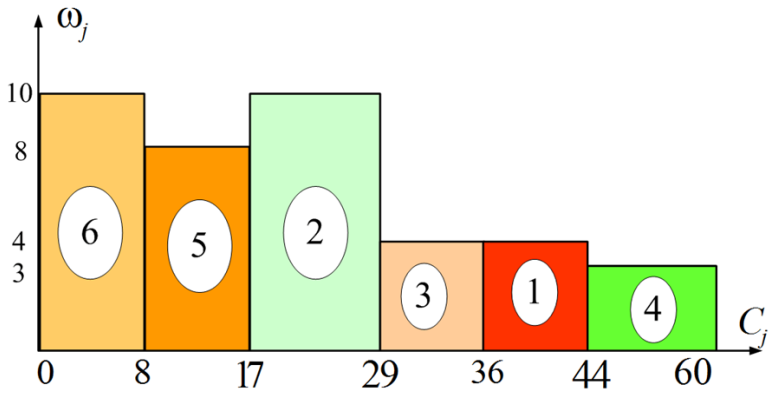

Figura 1.3: Representación de Gantt para la solución del problema clásico

Con esto se concluye que el modelo estaría resuelto, de precisarse que, el área que genera al disponer sobre la representación de Gantt del vector S con respecto al eje horizontal es el mínimo dentro de todo el conjunto de combinaciones posibles.

El tiempo de cálculo necesario para ordenar los trabajos de acuerdo con WSPT es el tiempo requerido para ordenar los trabajos de acuerdo con la relación de los dos parámetros. Un tipo simple puede hacerse en tiempo $O(nlog(n))$.

1.3.2. Problema $1|Prec|\sum \omega_j C_j$

Hay instancias cuando los trabajos tienen relaciones de precedencia, naturalmente surge la pregunta ¿cómo se ve afectada la minimización del tiempo de finalización total ponderado por las restricciones de precedencia?

Considere la forma más simple de restricciones de precedencia, es decir, las restricciones de precedencia que toman la forma de cadenas paralelas tal como se muestra en la figura 1.3.2.

Este problema todavía se puede resolver mediante un algoritmo relativamente

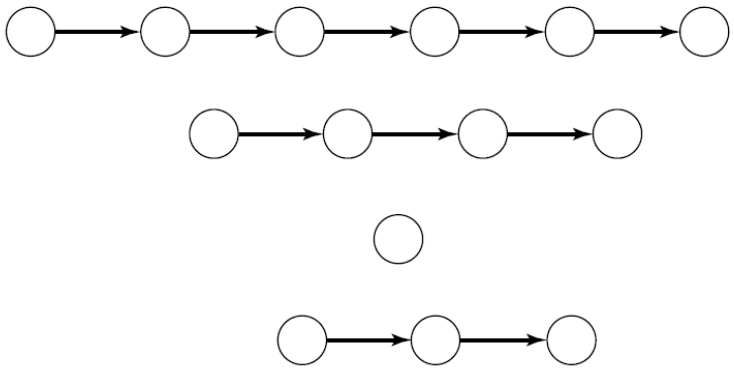

Figura 1.4: Restricción de precedencia en cadenas paralelas

simple y muy eficiente (tiempo polinomial). Este algoritmo se basa en algunas propiedades fundamentales de la programación con restricciones de precedencia.

Para expresar el método de resolución, consideremos dos cadenas de trabajos, la cadena 1 consiste en los trabajos $1, \cdots, k$ y la otra, cadena 2 consiste en los trabajos $k+1, \cdots, n$. Las restricciones de precedencia son como sigue:

$$1 \to 2 \to \cdots \to k$$

y

$$k+1 \to k+2 \to \cdots \to n.$$

El siguiente lema se basa en el supuesto de que si el planificador decide comenzar a procesar trabajos de una cadena, tiene que completar toda la cadena antes de que se le permita trabajar en puestos de la otra cadena. La pregunta es: si el programador desea minimizar el tiempo de finalización total ponderado de los n trabajos, ¿cuál de las dos cadenas debería procesar primero?

Lema 2 *Si*

$$\frac{\sum_{j=1}^{k}\omega_j}{\sum_{j=1}^{k}p_j} > (<)\frac{\sum_{j=k+1}^{n}\omega_j}{\sum_{j=k+1}^{n}p_j}$$

Entonces es óptimo para procesar la cadena de puestos de trabajo $1,\cdots,k$ *antes (después) de la cadena de los puestos de trabajo* $k+1,\cdots,n$.

Demostración. Realizaremos la prueba por contradicción. Con la secuencia $1,\cdots,k,k+1,\cdots,n$ el tiempo de culminación ponderada

$$\omega_1 p_1 + \cdots + \omega_k \sum_{j=1}^{k} p_j + \omega_{k+1} \sum_{j=1}^{k+1} p_j + \cdots + \omega_n \sum_{j=1}^{n} p_j$$

mientras que bajo la secuencia $k+1,\cdots,n,1,\cdots,k$,

$$\omega_{k+1} p_{k+1} + \cdots + \omega_n \sum_{j=k+1}^{n} p_j + \omega_1 \left(\sum_{j=k+1}^{n} p_j + p_1 \right) + \cdots + \omega_k \sum_{j=1}^{n} p_j$$

El tiempo de finalización total ponderado de la primera secuencia es menor que el total Tiempo de finalización ponderado de la segunda secuencia si

$$\frac{\sum_{j=1}^{k}\omega_j}{\sum_{j=1}^{k}p_j} > \frac{\sum_{j=k+1}^{n}\omega_j}{\sum_{j=k+1}^{n}p_j}$$

.

□

Una característica importante de la cadena

$$1 \to 2 \to \cdots \to k$$

es definida como sigue: sea $l*$ que satisface

$$\frac{\sum_{j=1}^{l*} \omega_j}{\sum_{j=1}^{l*} p_j} = \max_{1 \leq l \leq k} \left(\frac{\sum_{j=1}^{l} \omega_j}{\sum_{j=1}^{l} p_j} \right). \tag{1.6}$$

La relación en el lado izquierdo se denomina ρ-factor de la cadena $1, \cdots, K$ y se denota por $\rho(1, \cdots, k)$. El trabajo $l*$ se conoce como el trabajo que determina el factor ρ de la cadena.

Supongamos ahora que el planificador no tiene que completar todos los trabajos en una cadena antes de que se le permita trabajar en otra cadena. Puede procesar algunos trabajos de una cadena (respetando las restricciones de precedencia), cambiar a otra cadena y, en algún momento posterior, volver a la primera cadena. Si en el caso de cadenas múltiples, el tiempo de finalización total ponderado es la función objetivo, entonces se cumple el siguiente resultado.

Teorema 2 *Si el trabajo $l*$ determina $\rho\,(1, \cdots, k)$ entonces existe una secuencia óptima que procesa los trabajos $1, \cdots, l*$ uno tras otro sin interrupción por trabajos de otras cadenas.*

Demostración. Por contradicción. Supongamos que bajo la secuencia óptima el procesamiento de la subsecuencia $1, \cdots, l*$ es interrumpido por un trabajo, digamos trabajo v, de otra cadena. Es decir, la secuencia óptima contiene la subsecuencia $1, \cdots, u, v, u+1, \cdots, l*$ digamos una subsecuencia S. Basta con

demostrar que, o bien con subsecuencia $v, 1, \cdots, l*$, digamos S_1, o con subsecuencia $v, 1, \cdots, l*, v$, digamos S_2, el tiempo de finalización total ponderado es menor que con la subsecuencia S. Si no es menor con la primera subsecuencia, entonces tiene que ser menor con el segundo y viceversa.

Del lema anterior se deduce que si el tiempo de finalización total ponderado con S es menor que con S_1 entonces:

$$\frac{\omega_v}{p_v} < \frac{\omega_1 + \omega_2 + \cdots + \omega_u}{p_1 + p_2 + \cdots + p_u}.$$

Del lema anterior también se deduce que si el tiempo de finalización total ponderado con S es menor que con S_2 entonces:

$$\frac{\omega_v}{p_v} > \frac{\omega_{u+1} + \omega_{u+2} + \cdots + \omega_{l*}}{p_{u+1} + p_{u+2} + \cdots + p_{l*}}.$$

Si el trabajo $l*$ es el que determina el $\rho-$ factor de la cadena $1, \cdots, k$ entonces

$$\frac{\omega_{u+1} + \omega_{u+2} + \cdots + \omega_{l*}}{p_{u+1} + p_{u+2} + \cdots + p_{l*}} > \frac{\omega_1 + \omega_2 + \cdots + \omega_u}{p_1 + p_2 + \cdots + p_u}.$$

Si S es mejor que S_2 entonces

$$\frac{\omega_v}{p_v} > \frac{\omega_{u+1} + \omega_{u+2} + \cdots + \omega_{l*}}{p_{u+1} + p_{u+2} + \cdots + p_{l*}} > \frac{\omega_1 + \omega_2 + \cdots + \omega_u}{p_1 + p_2 + \cdots + p_u}.$$

Por tanto S_1 es mejor que S. El mismo argumento pasa si la interrupción de la

cadena es causada por más de un trabajo.

□

Ejemplo

Consideremos un problema en máquina simple con 7 trabajos tal como se muestra en la siguiente tabla

Trabajo	1	2	3	4	5	6	7
p_j	3	6	6	5	4	8	10
ω_j	6	18	12	8	8	17	10

y las cadenas de precedencia dadas por:

$$1 \to 2 \to 3 \to 4$$

y

$$5 \to 6 \to 7$$

Solución

Aplicaremos el método de la cadena como sigue. Por cada conjunto de trabajos encontraremos el valor del $\rho-$ factor $= \dfrac{\sum p_j}{\sum \omega_j}$

Primera iteración. Tomemos la cadena 1 dada por $\{1 \to 2 \to 3 \to 4\}$ y obtengamos los $\rho-$factores.

El $\rho-$ factor para la cadena 1

Trabajo	1	$1 \to 2$	$1 \to 2 \to 3$	$1 \to 2 \to 3 \to 4$
$\rho-$ factor	$\frac{3}{6}$	$\frac{9}{24}$	$\frac{15}{36}$	$\frac{20}{44}$
	0.5	0.375	0.416	0.455

El mínimo valor para el $\rho-$factor en la cadena 1 es $0{,}375$ y corresponde al conjunto de trabajos $1 \to 2$.

El $\rho-$ factor para la cadena 2

Trabajo	5	$5 \to 6$	$5 \to 6 \to 7$
$\rho-$ factor	$\frac{4}{8}$	$\frac{12}{25}$	$\frac{22}{43}$
	0.5	0.480	0.512

En este caso el mínimo valor del $\rho-$factor es $0{,}480$ y proviene del conjunto $5 \to 6$. Ahora procederemos a tomar la decisión en base al mínimo de ambas cadenas.

Comparación del $\rho-$ factor para la cadena 1 y cadena 2

Cadena	$\max \rho-$ factor	Conjunto de trabajos
Cadena 1	0.375	$1 \to 2$
Cadena 2	0.48	$5 \to 6$

En este caso sobre la base de todos los mínimos consideramos el menor conjunto de trabajos para ser incluidos en el schedule parcial de la siguiente forma $S = \{1, 2, \cdots\}$. Las nuevas cadenas quedarías definidas al eliminar los trabajos ya incluidos en el schedule parcial.

Segunda iteración

El $\rho-$ factor para la cadena 1

Conjunto de trabajos	3	$3 \to 4$
$\rho-$ factor	$\frac{6}{12}$	$\frac{11}{20}$
	0.5	0.55

Como la cadena 2 se mantuvo invariante procedemos a comparar.

Comparación del $\rho-$ factor para la cadena 1 y cadena 2

Cadena	$\max \rho-$ factor	Conjunto de trabajos
Cadena 1	0.5	3
Cadena 2	0.48	$5 \to 6$

En este caso se toma la decisión de incluir en el schedule parcial el conjunto de trabajos $5, 6$, obteniéndose

$$S = \{1, 2, 5, 6, \cdots\}.$$

Nuevamente estos trabajos son eliminados de las cadenas y se procede a repetir el proceso.

Tercera iteración.

Comparación del $\rho-$ factor para la cadena 1 y cadena 2

Cadena	$\max \rho-$ factor	Conjunto de trabajos
Cadena 1	0.5	3
Cadena 2	0.55	7

Sobre la base del mínimo se decide incluir en el schedule parcial al trabajo 3. Obteniendo

$$S = \{1, 2, 5, 6, 3, \cdots\}.$$

Cuarta iteración.

Comparación del $\rho-$ factor para la cadena 1 y cadena 2

Cadena	$\max \rho-$ factor	Conjunto de trabajos
Cadena 1	0.625	4
Cadena 2	0.55	7

Con la base sobre esta última comparación se determina el schedule

$$S = \{1, 2, 5, 6, 3, 7, 4\}$$

Capítulo 2

Aritmética Difusa

Esta sección está dedicada al estudio de los fundamentos de la aritmética difusa y algunos resultados importantes que son necesarios en la investigación.

2.1. Lógica Difusa

La teoría de la lógica difusa es un concepto relativamente nuevo, en comparación a la lógica clásica utilizada desde los albores del desarrollo del pensamiento matemático.

Creada por Lofti Zadeh en 1965, la Lógica Difusa aparece como un medio apropiado para procesar la información con imprecisión, es decir, la Lógica Difusa puede ser vista como un lenguaje que nos permite traducir sentencias sofisticadas del lenguaje natural, que contenga vaguedad o ambiguedad en un formalismo matemático mediante los conjuntos difusos, los cuales tienen un grado de pertenencia contínuo en el intervalo $[0, 1]$, esta idea es contraria a la teoría

clásica de conjuntos, en la que los objetos sólo tienen una grado de pertenencia (Función característica), que toman sólo el valor binario $\{0, 1\}$. A cada objeto x en un conjunto difuso X se le asigna un grado de pertenencia, cuya función usualmente se denota por $\mu(x)$ [63].

Muchas instancias que se encuentran en nuestro entorno físico, no tienen necesariamente una definición precisa del grado de pertenencia. Por ejemplo, que temperatura es "bastante alta", que "velocidades son bajas", etc. Sin embargo, el hecho es que dichas clases definidas de manera ïmprecisa"juegan un papel importante en las desiciones que toma el ser humano, particularmente en el área de optimización.

2.1.1. Subconjuntos difusos

En la Lógica Difusa los conjuntos se definen por sus respectivas funciones de pertenencia.

En la lógica convencional, supongamos que A es un subconjunto de S definido como un mapeo de los elementos de S a los elementos del conjunto $\{0, 1\}$, representados como un conjunto de pares ordenados con exactamente un par ordenado presente por cada elemento de S, el segundo elemento del par ordenado es un elemento del conjunto $\{0, 1\}$. El 0 representa la no-pertenencia y el 1 la pertenencia.

Al encontrar la siguiente expresión:

x está en A.

se evalúa su valor de verdad encontrando el par ordenado cuyo primer elemento es x, si el segundo elemento del par ordenado es 1 entonces la oración es verdadera, en cambio si es 0 entonces es falsa. Por lo tanto el elemento pertenece o no pertenece al conjunto A.

Si hacemos una comparación con conjuntos difusos, en lugar del conjunto {0,1}, hablaríamos del intervalo $[0, 1]$. El valor 0 denotaría una no-pertenencia completa, por el contrario el valor 1 denotaría una pertenencia completa y cualquier otro valor intermedio determinaría grados intermedios de pertenencia.

Cabe aclarar que los grados difusos no son equivalentes a los porcentajes manejados en probabilidad. Las medidas de probabilidad establecen si algo pasará o no. Lo difuso establece el grado mediante el cual algo ocurrirá o la existencia de una condición.

De lo anterior se desprende la definición de conjunto difuso [36].

Sea X un conjunto cuyos elementos se denotan por x, y sea A un subconjunto de X . La pertenencia de un elemento x de X al conjunto A viene dada por la función característica.

$$\mu_A(x) = \begin{cases} 1 & \text{si y sólo si } x \in A \\ 0 & \text{si y sólo si } x \notin A \end{cases} \tag{2.1}$$

$\{1, 0\}$ es el llamado conjunto valoración (rango).

Si el conjunto valoración es el intervalo real $[0, 1]$, A se denomina un conjunto difuso [36] y $\mu_A(x)$ mide el grado de pertenencia del elemento x al conjunto A.

Un conjunto difuso A se caracteriza por el conjunto de pares

$$A = \{(x, \mu_A(x)), x \in X)\}$$

Definición 1 *Sea X un conjunto universal clásico, A es llamado un conjunto difuso en X, si definimos una función $\mu_A(x) \to [0, 1]$ denominada función de pertenencia, tal que a cada elemento x de X le asocia un número $\mu_A(x)$ entre 0 y 1, denominado el grado de pertenencia de x en A.*

2.1.2. Operaciones básicas entre conjuntos difusos

Resumiremos algunas operaciones básicas entre conjuntos difusos, las cuales usaremos en este trabajo.

Inclusión.- Sean A y B dos subconjuntos difusos en X, entonces A está incluido en B si y sólo si

$$\mu_A(x) \leq \mu_B(x) \, \forall x \in X$$

.

La inclusión se denotará por $A \subset B$.

Igualdad.- Los conjuntos difusos A y B son llamados iguales si y sólo si

$$\mu_A(x) = \mu_B(x)$$

.

Se denotará la igualdad por $A = B$.

Complemento.- Los conjuntos difusos A y B en X son complementarios si y sólo si

$$\mu_A(x) = 1 - \mu_B(x) \, \forall x \in X$$

.

El complemento de A se denota por $B = A^c$, ó $A = B^c$ donde A^c y B^c son complementos de A y B respectivamente.

Intersección.- La intersección de dos conjuntos difusos A y B en X está definida por:

$$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, \forall x \in X.$$

Unión.- La unión de dos conjuntos difusos A y B en X está definida por:

$$\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\}, \forall x \in X.$$

Diferencia.- La diferencia $A - B$ de los conjuntos difusos A y B en X es caracterizada por

$$\mu_{A \cap B^c}(x) = min\{\mu_A(x), \mu_{B^c}(x)\}, \forall x \in X.$$

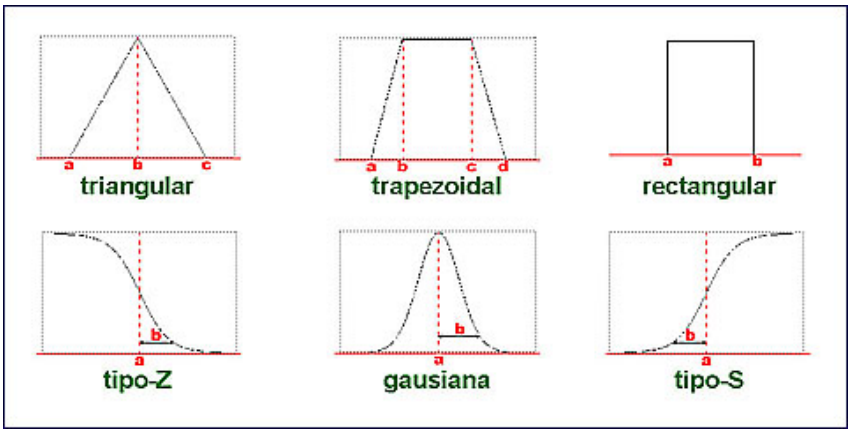

Figura 2.1: Algunos tipos de funciones de pertenencia.

donde $\mu_{B^c}(x)$ es el complemento de B, $\mu_{B^c}(x) = 1 - \mu_B(x)$

2.1.3. Determinación de la función de pertenencia

Una buena modelación de un concepto e expresión linguistica mediante un conjunto difuso, depende de la forma en que se determine su función de pertenencia. Por ello se debe tener cuidado en construir una función de pertenencia precisa y justificable. Los métodos empíricos son los métodos mas utilizados y se basan en datos que se tengan del problema a resolver.

Existen diversos métodos para medir la subjetividad de los grados de pertenencia a la clase que se quiere modelar, entre estos, se tiene.

- Evaluación subjetiva: una persona asigna un grado de pertenencia subjetivo a cada elemento; normalmente esta evaluación la realizan expertos en el tema o problema de que se trate.

- Frecuencias o probabilidades: estadísticas basadas en histogramas.
- Funciones ad-hoc: Se suele utilizar un pequeño conjunto de funciones sencillas (por ejemplo funciones triangulares o en forma de trapecio) como funciones de pertenencia. De este modo, el problema se reduce a la elección de unos pocos parámetros en dichas funciones.

Un resumen de los principales métodos de construcción de funciones de pertenencia puede encontrase en ([60], [37]). Como construir e interpretar las funciones de pertenencia es un tema que aún sigue siendo estudiado [**?**].

Klir y Yuan (1995) [37], sostienen que el problema de la construcción de funciones de pertenencia que capturen adecuadamente los significados de los términos lingüísticos empleados en una aplicación particular, así como la determinación del significado de las operaciones asociadas a los términos lingüísticos, no son problemas de la teoría de conjuntos difusos, sino que pertenecen a un área más general de problemas de adquisición de conocimiento.

El escenario en el cual se desarrolla la construcción de un conjunto difuso incluye un dominio de conocimiento específico, con uno o más expertos que dominen el tema, y un dominio de conocimiento de las técnicas necesarias para construir las funciones de pertenencia, y sólo la interacción de ambos permitirá extraer conocimiento.

El gráfico de una función de pertenencia puede ser triangular, trapezoidal,

sigmoide, cuadrático, etc. (ver Figura 2.1).

Las funciones de pertenencia también se pueden construir mediante el empleo de redes neuronales, algoritmos genéticos, algoritmos de búsqueda tabú, etc. Este tipo de métodos se emplean especialmente en el diseño de controladores difusos.

Es importante revisar el concepto de números difusos como una extensión propia de los conjuntos difusos y las respectivas operaciones que se pueden realizar entre ellos, puesto que muchas de las situaciones reales, sobre todo en situaciones donde hay que tomar decisiones.

2.2. Números difusos

2.2.1. Definiciones Generales

Definición 2 : [10] Un número difuso $\widetilde{A}$ es un conjunto $\mu_{\widetilde{A}}$ de la recta real, convexo, normalizado y tal que:

a) $\exists x_0 \in \mathbb{R} / \mu_A(x_0) = 1$ que suele llamarse moda, y

b) μ_A es continua a trozos.

Así todo número difuso está caracterizado por una función de pertenencia

$$\mu_{\widetilde{A}} : \mathbb{R} \to [0, 1]$$

y toda función como la anterior genera un número difuso donde,$\forall x \in \mathbb{R}$, $\mu_A(x)$ es el grado de pertenencia de x al número difuso $\widetilde{A}$.

$$\mu_{\widetilde{A}}(x) = \begin{cases} 0, & x < a \\ L(x), & a \leq x < c \\ 1, & c \leq x \leq d \\ R(x), & d < x \leq b \\ 0, & x > b \end{cases} \tag{2.2}$$

Donde $L(x)$ representa una función semicontinua, estrictamente creciente para $a \leq x < c$, en el cual exite a tal que $L(x) = 0$ para $x < a$ y $R(x)$ otra función estrictamente decreciente para $d < x \leq b$, en el cual también existe b tal que $R(x) = 0$ para $x \geq b$. La representación de estas funciones de referencia viene dada en la figura

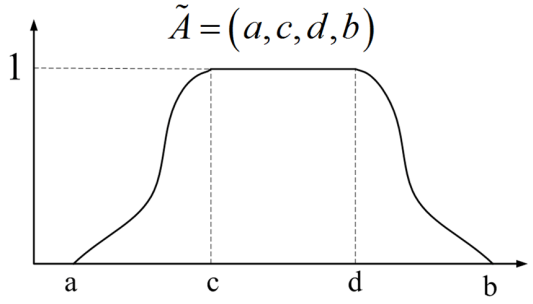

Figura 2.2: Número difuso

2.3. Ordenamiento de números difusos

La comparación de dos números difusos permite deducir de entre dos números difusos $\widetilde{A}$ y $\widetilde{B}$ cuál es mayor, pero los números difusos no siempre proporcionan

un conjunto completamente ordenado como lo hacen los números reales. Todos los métodos de comparación tienen alguna desventaja, la elección de un método depende del decisor de acuerdo a sus propósitos en cualquier caso de estudio, a continuación mencionaremos algunos métodos de comparación basados en [**?**].

2.3.1. Métodos basados en la definición de una función ordenadora

Sea F($\mathbf{R}$) el conjunto de funciones de pertenencia sobre $\mathbf{R}$ y consideremos $\widetilde{A}$,$\widetilde{B} \in F(\mathbf{R})$. Un método de comparación entre ellos consiste en la definición de una función $f : F(\mathbf{R}) \to \mathbf{R}$. Si se conoce esta función f entonces:

$$f\left(\widetilde{A}\right) < f\left(\widetilde{B}\right) \Leftrightarrow \widetilde{A} < \widetilde{B}$$
$$f\left(\widetilde{A}\right) > f\left(\widetilde{B}\right) \Leftrightarrow \widetilde{A} > \widetilde{B}$$
$$f\left(\widetilde{A}\right) = f\left(\widetilde{B}\right) \Leftrightarrow \widetilde{A} = \widetilde{B}$$

Funciones de ordenación lineal

Definición 3 *Una función f es de ordenación lineal, si:*

A) $\forall \widetilde{A}, \widetilde{B} \in F(\mathbf{R})$ *,* $f\left(\widetilde{A} + \widetilde{B}\right) = f\left(\widetilde{A}\right) + f\left(\widetilde{B}\right)$

B) $\forall \widetilde{A} \in F(\mathbb{R}), \forall r \in \mathbb{R}, r > 0$*,* $f\left(r \cdot \widetilde{A}\right) = r \cdot f\left(\widetilde{A}\right)$

Definiremos algunas funciones de ordenación lineal.

A) Primer índice de Yager

Sea $\widetilde{A} \in F(\mathbb{R})$, este índice queda definido mediante

$$f_1\left(\widetilde{A}\right) = \frac{\int_S h(x) \cdot \mu_{\widetilde{A}}(x)dx}{\int_S \mu_{\widetilde{A}}(x)dx} \tag{2.3}$$

Donde $h(x)$ es una medida de importancia del valor x y S es el soporte de $\widetilde{A}$. Si $h(x) = x$, este índice representa la abscisa del centro de gravedad del número difuso $\widetilde{A}$.

El valor de este índice en el caso de un número difuso de forma triangular $\widetilde{A} = (a_1, a_2, a_3)$ es:

$$f_1(\widetilde{A}) = \frac{a_1 + a_2 + a_3}{3} \tag{2.4}$$

En consecuencia, para comparar dos números difusos $\widetilde{A}$ y $\widetilde{B}$ de forma triangular diremos que $\widetilde{A} \leq \widetilde{B}$ cuando

$$f_1\left(\widetilde{A}\right) \leq f_1\left(\widetilde{B}\right).$$

B)Tercer índice de Yager Sea $\widetilde{A} \in F(\mathbb{R})$, este índice se define como

$$f_3\left(\widetilde{A}\right) = \int_0^1 M(A_\alpha) \cdot d\alpha \tag{2.5}$$

donde A_α es el $\alpha-$corte de $\widetilde{A}$ y $M(A_\alpha)$ es el valor medio de los elementos de A_α. Si el número difuso es de forma triangular $\widetilde{A} = (a_1, a_2, a_3)$ entonces se tiene que

$$A_\alpha = [a_1 + \alpha(a_2 - a_1), a_3 - \alpha(a_3 - a_2)]$$

$$M(A_\alpha) = \frac{(a_1 + a_3) + \alpha(2a_2 - a_1 - a_3)}{2}$$

luego el valor del índice para un número difuso triangular es

$$f_{\widetilde{A}} \frac{a_1 + 2a_2 + a_3}{4}. \tag{2.6}$$

Así pues para dos números difusos triangulares $\widetilde{A} = (a_1, a_2, a_3)$ y $\widetilde{B} = (b_1, b_2, b_3)$, decimos que $\widetilde{A} \leq \widetilde{B}$ cuando $f_3\left(\widetilde{A}\right) \leq f_3\left(\widetilde{B}\right)$ es decir si:

$$a_1 + 2a_2 + a_3 \leq b_1 + 2b_2 + b_3 \tag{2.7}$$

C)Relación de Adamo Utilizando el concepto de $\alpha-$corte, Adamo define un índice de $\alpha-$ preferencias, dado por:

$$f_\alpha\left(\widetilde{A}\right) = \text{máx}\left\{x/\mu_{\widetilde{A}}(x) \geq \alpha\right\}. \tag{2.8}$$

donde $\alpha \in [0, 1]$ es un grado que el decisor fija a priori de acuerdo a sus propios intereses. Para un número difuso triangular $\widetilde{A} = (a_1, a_2, a_3)$ se tendrá que:

$$f_{\widetilde{A}}(\alpha) = a_3 - (a_3 - a_2)\alpha. \tag{2.9}$$

Utilizando este índice diremos que dos números difusos, del tipo triangular, verifican $\widetilde{A} \leq \widetilde{B}$ en un grado α cuando:

$$a_3 - (a_3 - a_2)\alpha \leq b_3 - (b_3 - b_2)\alpha \tag{2.10}$$

Funciones de ordenación no lineal

D)Segundo índice de Yager El segundo índice que Yager propone es la medida de consistencia de un número difuso cuyo soporte está contenido en el intervalo $[0, 1]$ con el conjunto difuso lineal

$$\widetilde{A} = \left\{\left(x, \mu_{\widetilde{A}(x)}\right)/\mu_{\widetilde{A}}(x) = x\right\}$$

es decir

$$f_2\left(\widetilde{A}\right) = \underset{x \in S}{\text{máx}} \text{mín}\left\{x, \mu_{\widetilde{A}(x)}\right\} \tag{2.11}$$

El valor de este índice para un número difuso $\widetilde{A}$ de forma triangular, tal que $[a_1, a_3] \subseteq [0, 1]$ es

$$f_2\left(\widetilde{A}\right) = \frac{a_3}{a_3 - a_2 + 1} \tag{2.12}$$

Por tanto en la comparación de dos números se tiene $\widetilde{A} \leq \widetilde{B}$ cuando

$$\frac{a_3}{a_3 - a_2 + 1} \leq \frac{b_3}{b_3 - b_2 + 1}$$

2.3.2. Método basado en criterio del Intervalo Esperado

Según Jiménez en [34], sean $\widetilde{A}$ y $\widetilde{B}$ dos números difusos, el grado en el cual $\widetilde{A}$ es mayor o igual que $\widetilde{B}$ viene dado por

$$M(\widetilde{A}, \widetilde{B}) = \begin{cases} 0 & \text{si } E_2^A - E_1^B < 0 \\ \dfrac{E_2^A - E_1^B}{E_2^A - E_1^B - \left(E_1^A - E_2^B\right)} & \text{si } 0 \in \left[E_1^A - E_2^B, E_2^A - E_1^B\right] < 0 \\ 1 & \text{si } E_2^A - E_1^B > 0 \end{cases} \tag{2.13}$$

donde $\left[E_1^A, E_2^A\right]$ y $\left[E_1^B, E_2^B\right]$ son los intervalos esperados correspondientes a los números difusos $\widetilde{A}$ y $\widetilde{B}$. Además, cuando $MM(\widetilde{A}, \widetilde{B}) = 0{,}5$ se dice que $\widetilde{A}$ y $\widetilde{B}$ son indiferentes.

2.3.3. Métodos basado en la relación de preferencias

Dubois y Prade en [19] proponen índices que describen la posición relativa de dos números difusos.

Grado de posibilidad de dominancia de $\widetilde{A}$ sobre $\widetilde{B}$

Se define como

$$Pos\left(\widetilde{A} \geq \widetilde{B}\right) = \sup_{x,y/x \geq y} \text{mín}\left[\mu_{\widetilde{A}}(x), \mu_{\widetilde{B}}(y)\right] \tag{2.14}$$

Así, dados dos números fuzzy con funciones de pertenencia de tipo triangular, diremos que $\widetilde{A} \leq \widetilde{B}$ si

$$Pos\left(\widetilde{A} \leq \widetilde{B}\right) \geq Pos\left(\widetilde{A} \geq \widetilde{B}\right) \tag{2.15}$$

$$\mathcal{R}(\widetilde{A}, \widetilde{B}) = Pos\left(\widetilde{A} \leq \widetilde{B}\right) = \begin{cases} 0 & \text{si } b_3 \leq a_1 \\ \dfrac{b_3 - a_1}{b_3 - b_2 + a_2 - a_1} & \text{si } a_2 > b_2 \wedge b_3 > a_1 \\ 1 & \text{si } a_2 \leq b_2 \end{cases} \tag{2.16}$$

$$\mathcal{R}(\widetilde{B}, \widetilde{A}) = Pos\left(\widetilde{B} \leq \widetilde{A}\right) = \begin{cases} 0 & \text{si } a_3 \leq b_1 \\ \dfrac{a_3 - b_1}{a_3 - a_2 + b_2 - b_1} & \text{si } b_2 > a_2 \wedge a_3 > b_1 \\ 1 & \text{si } b_2 \leq a_2 \end{cases} \tag{2.17}$$

Por tanto $\widetilde{A} \leq \widetilde{B}$ si

$$\frac{b_3 - a_1}{b_3 - b_2 + a_2 - a_1} \geq \frac{a_3 - b_1}{a_3 - a_2 + b_2 - b_1} \tag{2.18}$$

Grado de necesidad de dominancia de $\widetilde{A}$ sobre $\widetilde{B}$

Se define como

$$Nec\left(\widetilde{A} \geq \widetilde{B}\right) = \inf_{x} \sup_{y/y \leq x} \text{máx}\left[1 - \mu_{\widetilde{A}}(x), \mu_{\widetilde{B}}(y)\right] \tag{2.19}$$

Considerando dos números fuzzy con funciones de pertenencia de tipo triangular, se obtienen los siguientes grados de necesidad

$$\mathcal{R}(\widetilde{A},\widetilde{B}) = Nec\left(\widetilde{A} \leq \widetilde{B}\right) = \begin{cases} 0 & \text{si } b_2 \leq a_1 \\ \dfrac{b_2 - a_1}{a_2 - a_1 + b_2 - b_1} & \text{si } b_2 > a_1 \wedge a_2 > b_1 \\ 1 & \text{si } a_2 \leq b_1 \end{cases} \tag{2.20}$$

$$\mathcal{R}(\widetilde{B},\widetilde{A}) = Nec\left(\widetilde{B} \leq \widetilde{A}\right) = \begin{cases} 0 & \text{si } a_2 \leq b_1 \\ \dfrac{a_2 - b_1}{a_2 - a_1 + b_2 - b_1} & \text{si } a_2 > b_1 \wedge b_2 > a_1 \\ 1 & \text{si } b_2 \leq a_1 \end{cases} \tag{2.21}$$

Por tanto $\widetilde{A} \leq \widetilde{B}$ si $Nec\left(\widetilde{A} \leq \widetilde{B}\right) \geq Nec\left(\widetilde{B} \leq \widetilde{A}\right)$ que se reduce a

$$a_1 + a_2 \leq b_1 + b_2 \tag{2.22}$$

2.4. Intervalo esperado difuso

A medida que la teoría propuesta por Zadeh en [63] sobre los conjuntos difusos fue implementándose, surgió la necesidad de extender los principios, elementos que sustentan y formulan la teoría matemática y estadística al entorno fuzzy. Dado que un número difuso constituye en sí una distribución de posibilidad, surgen naturalmente la idea de extender el concepto de probabilidad difusa, esperanza difusa, entre otros. En esta sección se presenta el desarrollo de un elemento importante en cuanto a las medidas de comparación para números difusos basados en el Intervalo esperado de un número difuso. El análisis que se presenta se basa en lo elaborado por Heilpern en [30].

Yager en [62], presenta una definición para la probabilidad de un subconjunto fuzzy en un determinado espacio de probabilidad a través de la integral de

Lebesgue-Stieljes. Al respecto, la probabilidad difusa es vista como un subconjunto del intervalo unitario, el cual toma valores lingüísticos como altamente probable, probable, etc.

DIversas han sido las propuestas para determinar el valor medio que resume el comportamiento de una variable aleatoria fuzzy, Dubois y Prade en [20], presentan este concepto a través del grado de posibilidad y necesidad. Heilpern en [30] unifica los conceptos de posibilidad y necesidad, a través de la familia de intervalos aleatorios generados por un número difuso.

En el caso de conjuntos fuzzy, para Schneider y Kandel [56] el valor esperado difuso es una medida de tipicalidad, es decir consiste en encontrar el valor típico de una población. Dicho valor puede ser o bien el peso de un grupo dentro de una determinada población, o el grado de membresía del grupo en dicha población.

2.4.1. Nociones previas

En virtud de esto, es necesario definir el intervalo aleatorio y entender la funcionalidad y reciprocidad de la funcióon de distribución de probabilidad con el número fuzzy.

En [29], se considera el problema de generar un intervalo aleatorio por un conjunto fuzzy. Un conjunto aleatorio dado un espacio universal X es un mapeo medible del espacio de probabilidad (Ω, A, P) a la familia $\mathcal{P}(X)$ de todos los subconjuntos de X. Bajo esta definición, un intervalo aleatorio no es más que un

conjunto aleatorio cuya imagen bajo el mapeo medible coincide con la clase de todos los intervalos cerrados de la recta real. Tomando $X = \mathbb{R}$ se puede inducir una nueva clase de intervalos cerrados de la recta real $S(\omega)$ como sigue

$$S(\omega) = [a(\omega), b(\omega)] \tag{2.23}$$

donde $\omega \in \Omega$ y $a(\omega) \leq b(\omega)$, que a su vez es elemento del espacio

$$\mathbb{R}^2 = \{(x; y) : x, y \in \mathbb{R}, x \leq y\}. \tag{2.24}$$

Un conjunto aleatorio S, según [30], se puede caracterizar en términos de las probabilidades inferior P_* y superior P^* definidas por:

$$P_* = Pr(S \subset B), \quad P^* = Pr(B \cap S \neq \Phi) \tag{2.25}$$

que no son otra cosa más que el grado de posibilidad y necesidad para un conjunto difuso S.

Con la ecuación 2.25, Dubois y Prade [20], generan las funciones de distribución inferior y superior

$$F_* = P_*((-\infty, x]), \quad F^* = P^*((-\infty, x]) \tag{2.26}$$

las cuales verifican $\forall x \in \mathbb{R}$,

$$F_*(x) \leq F(x) \leq F^*(x) \tag{2.27}$$

es decir una función de distribución $F(x)$ está limitada por las funciones de distribución superior e inferior para un intervalo $(-\infty, x]$. Este resultado es muy importante

puesto que si se quiere definir el valor esperado de un intervalo cerrado es necesario calcular el valor esperado de ambas las distribuciones superior e inferior, [20].

Dempster's [17] propone el valor medio para un intervalo fuzzy semicontinua superior Q, denotado por $E(Q)$ como el conjunto de números

$$E\left(Q\right)=\left\{E(P)\mid P\in\mathcal{P}\left(Q\right)\right\}$$

donde $E(P)$ es el valor esperado usual

$$E\left(P\right)=\int_{-\infty}^{\infty}x\,\mathrm{d}F(x) \tag{2.28}$$

con F como una distribución de P.

Con estas últimas ideas Dubois [20], define los valores medios superior e inferior y en concordancia con [17]

$$E_{*}\left(Q\right)=\int_{-\infty}^{\infty}x\,\mathrm{d}F^{*}(x);\quad E^{*}\left(Q\right)=\int_{-\infty}^{\infty}x\,\mathrm{d}F_{*}(x). \tag{2.29}$$

2.4.2. Formación del Intervalo Esperado

El número difuso A induce una clase de conjuntos aleatorios $\mathcal{L}\left(A\right)$, [25], el cual satisface la condición

$$\mathrm{Pr}\left(x\in S\right)=\mu_{A}\left(x\right) \tag{2.30}$$

para $S\in\mathcal{L}\left(A\right)$. Se debe aclarar que, cuando se tiene un conjunto aleatorio éste puede generar un conjunto difuso caracterizado por una asignación básica de probabilidad, aunque se debe considerar que muchos y diferentes conjuntos aleatorios pueden generar el mismo conjunto difuso.

El intervalo aleatorio $S(A)$ es generado por todos los cortes del número difuso A ([29]), esto es los números difusos inducen una clase de intervalos aleatorios $RI\,(A)$, [30].

A continuación se presenta un lema importante para la definición y cálculo del intervalo esperado.

Lema 3 . [30]. *Sea $S \in RI\,(A)$, entonces se cumple:*

i. $f_A\,(z) = F\,(z, b)$ *y* $g_A\,(z) = 1 - F\,(c, z)$

ii. $F_*\,(z) = F\,(c, z)$ *y* $F^*\,(z) = F\,(z, b)$

donde F es una función de distribución de S.

Como se observa el número difuso induce de manera natural y de acuerdo a su definición las funciones de distribución.

Definición 4 . [30]. *El valor esperado del intervalo aleatorio $S \in RI\,(A)$ es llamado el valor esperado de un número difuso y se denota por* ***EI(A)****.*

Por ende, como todo número difuso A induce una clase de intervalos aleatorios se tiene

$$F^* = f_A\,(x)\,,\ F_* = 1 - g_A\,(x)$$

con lo cual, un procedimiento de cálculo para el intervalo esperado se toma de (2.29) y se formula como:

$$\begin{aligned} \mathsf{EI}\,(A) &= [\mathsf{Es}_1,\ \mathsf{Es}_2] && (2.31)\\ &= \left[\int_a^c x \mathsf{dF}^*\,(x)\,,\ \int_d^b x \mathsf{dF}_*\,(x)\right] && (2.32)\\ &= \left[\int_a^c x \mathsf{d} f_A\,(x)\,,\ -\int_d^b x \mathsf{d} g_A\,(x)\right] && (2.33) \end{aligned}$$

Vamos a calcular los extremos del intervalo esperado, es decir desarrollemos las integrales escritas en (2.33) por partes y por definición de (**??**):

$$\begin{aligned} Es_1 &= \int_a^c x\mathsf{d}f_A(x) = xf_A(x)|_a^c - \int_a^c f_A(x)\,\mathsf{d}x \\ &= cf_A(c) - af_A(a) - \int_a^c f_A(x)\,\mathsf{d}x \\ \Rightarrow Es_1 &= c - \int_a^c f_A(x)\,\mathsf{d}x. \end{aligned} \tag{2.34}$$

Similarmente para el cálculo de ES_2 se tiene

$$\begin{aligned} Es_2 &= -\int_d^b x\mathsf{d}g_A(x) = \int_d^b g_A(x)\,\mathsf{d}x - xg_A(x)|_d^b \\ &= \int_d^b g_A(x)\,\mathsf{d}x - bg_A(b) + dg_A(d) \\ \Rightarrow Es_2 &= d + \int_d^b g_A(x)\,\mathsf{d}x. \end{aligned} \tag{2.35}$$

2.4.3. Un parámetro para el valor esperado

Para efectos de programación y facilidad en la manipulación de este concepto puede adecuarse un cambio de variable para Es_1 y Es_2, siempre que se garantice que A es un número difuso con funciones de membresía continua. Este cambio se determina por

$$t = f_A(x) \rightarrow x = f_A^{-1}(x) \ \wedge \ \mathsf{d}x = \mathsf{d}f_A^{-1}(t)$$

los límites de integración se determinan para cada uno de los dominios, en el caso de $f_A(x)$ se tiene:

$$a \leq x \leq c \Rightarrow 0 \leq t \leq 1$$

entonces Es_1 se reduciría mediante:

$$\begin{aligned} Es_1 &= c - \int_a^c f_A(x)\,\mathrm{d}x = c - \int_0^1 t\mathrm{d}f_A^{-1}(t) \\ &= c - tf_A^{-1}(t)\,|_0^1 + \int_0^1 f_A^{-1}(t)\,\mathrm{d}t \\ Es_1 &= \int_0^1 f_A^{-1}(t)\,\mathrm{d}t \end{aligned} \tag{2.36}$$

Tal y como se puede observar, para el cálculo del intervalo esperado solo es necesario conocer $\mu_A(x)$ y esta definición no depende del miembro de la clase de RI(A). Lo mencionado se enuncia en el siguiente teorema.

Teorema 3 . [30]. *Si S y T* $\in RI(A)$*, entonces EI*$(S) =$ *EI*(T).

El intervalo esperado puede ser utilizado como una medida de comparación entre números difusos.

Definición 5 . [30] *El valor esperado de un número difuso A es el centro del intervalo esperado asociado a dicho número , se denota por EV*(A) *y se calcula por:*

$$EV(A) = \frac{1}{2}(Es_1 + Es_2) \tag{2.37}$$

Si esta medida de comparación para un determinado intervalo cerrado se compara con otras aproximaciones para un número difuso, hay que mencionar la equivalencia entre los conceptos desarrollado por Dubois y Heilpern, esta equivalencia se enuncia en el siguiente corolario.

Corolario 2 . [30]. *El intervalo esperado de un número difuso* A *es igual al valor medio de este número, esto es,*

$$EI(A) = E(A) \tag{2.38}$$

Las funciones lineales, fácilmente verifican las condiciones puestas para $f_A(x)$ y $g_A(x)$, con lo que ahora consideraremos el caso de un número difuso triangular (TFN) el cual es dado por la siguiente función de membresía:

$$\mu_A(x) = \begin{cases} \dfrac{x-a}{b-a}, & \text{si } a \leq x < b \\ \dfrac{x-d}{b-d}, & \text{si } b \leq x \leq d \\ 0, & \text{otros casos} \end{cases} \tag{2.39}$$

Se puede hacer uso de (2.34) y (2.35) para calcular los extremos del intervalo esperado de un TFN.

$$\begin{aligned} Es_1 &= b - \int_a^b f_A(x)\,\mathrm{d}x \\ &= b - \int_a^b \left(\frac{x-a}{b-a}\right)\mathrm{d}x \\ &= b - \frac{x^2}{2(b-a)}\Big|_a^b + \frac{ax}{(b-a)}\Big|_a^b \\ \Rightarrow Es_1 &= \frac{a+b}{2} \end{aligned} \tag{2.40}$$

procedemos a calcular el extremo superior del intervalo esperado:

$$\begin{aligned} Es_2 &= b + \int_b^d g_A(x)\,\mathrm{d}x \\ &= b + \int_b^d \left(\frac{x-d}{b-d}\right)\mathrm{d}x \\ &= b + \frac{x^2}{2(b-d)}\Big|_b^d + \frac{dx}{(b-d)}\Big|_b^d \\ \Rightarrow Es_2 &= \frac{b+d}{2} \end{aligned} \tag{2.41}$$

Recogiendo los últimos resultados en una sola expresión tenemos que para un número difuso triangular, el intervalo esperado viene dado por

$$\mathrm{EI}(A) = \left[\frac{a+b}{2}, \frac{b+d}{2}\right] \tag{2.42}$$

La ecuación (2.42) muestra que para un número difuso triangular,los extremos del intervalo esperado están determinados por el promedio de los extremos de la recta que contienen las funciones que definen al número difuso triangular.

Teniendo en cuenta (2.37), el valor esperado para esta clase de números difusos está determinado por:

$$\text{EV}(A) = \frac{a+2b+d}{2} \tag{2.43}$$

Ejemplo numérico

Considere el número difuso triangular A con la siguiente función de membresía:

$$\mu_A(x) = \begin{cases} x-2, & \text{si } 2 \leq x < 3 \\ -x+4, & \text{si } 3 \leq x \leq 4 \\ 0, & \text{otros casos} \end{cases} \tag{2.44}$$

Para el cálculo del EI, podemos hacer uso de la ecuación 2.43 con lo cual se obtendría para $a = 2, b = 3 \wedge d = 4$

$$\begin{aligned} \text{EI}(A) &= \left[\frac{2+3}{2}, \frac{3+4}{2}\right] \\ \text{EI}(A) &= [2{,}5, 7{,}5] \end{aligned} \tag{2.45}$$

Puede verse en el último resultado que el intervalo esperado es un subconjunto de aquel intervalo que define al número difuso, en otras palabras corresponde a un determinado $\alpha-$corte.

El valor esperado para este número difuso es:

$$\text{EV}(A) = \frac{2+2(3)+4}{2} = 6. \tag{2.46}$$

Capítulo 3

Métodos y modelos en Fuzzy Single Machine

3.1. Estado del arte en Fuzzy Scheduling

Hablar sobre problemas de scheduling implica involucrarse en el campo de la optimización combinatoria y de los problemas con satisfacción de restricciones. En lo que se refiere a teorías de la computación estos problemas corresponden al grupo de NP-duros: es decir aquellos en que, si bien la mayor parte de metodologías trabajadas son solo aproximaciones, adquiere mayor complejidad a medida en que se incluyen más variables y restricciones, lo cual da lugar a la incorporación de algoritmos que incluyen tiempos computacionales muy elevados.

Dentro del proceso de toma de decisiones en actividades industriales, el scheduling juega un papel muy importante, pues hay una gran variedad de aspectos que deben considerarse al momento de elaborar un modelo para estos problemas. La mayor parte de los trabajos desarrollados en esta área se basan en modelos

de optimización determinísticos cuyos parámetros considerados son conocidos. En años recientes este paradigma ha ido cambiando, pues en la elaboración de modelos de scheduling han de considerarse a la imprecisión como un elemento importante y también a la caracterización fundamental. Como escriben Li e Ierapetrirou en [42], en las plantas reales, la incertidumbre es una preocupación muy importante que se acopla a los procesos de scheduling. Así, muchos de los parámetros asociados al scheduling no se conocen con exactitud. Estos parámetros, como la disponibilidad de materias primas, los precios, la fiabilidad de la máquina, y las exigencias del mercado, varían con respecto al tiempo y están, a menudo sujetos a desviaciones inesperadas. En síntesis, la incertidumbre juega el papel de validar el uso de los modelos matemáticos y de preservar la viabilidad de las plantas así como su eficiencia durante las operaciones.

Existen dos causas importantes para la aparición de la incertidumbre: la aleatoriedad y la difusidad. Si bien es cierto, la incertidumbre en los procesos de scheduling se describe mediante modelos probabilísticos, la teoría de conjuntos difusos se aplica para optimizar el scheduling usando técnicas de búsqueda heurística. La diferencia principal radica en cómo se realiza el modelamiento. Si la incertidumbre radica especialmente en la imprecisión que presentan las variables, entonces lo recomendable es formular los problemas de scheduling usando las técnicas para lógica y la aritmética difusa. La idea de representar las variables por números difusos radica en su utilidad para el tratamiento de la imprecisión o en la representación de un conjunto de valores posibles. En este caso el problema se convierte en un problema denominado Fuzzy Scheduling.

3.1.1. Parámetros difusos en problemas de Scheduling Machine

En esta sección se revisan algunos de los parámetros que intervienen en un problema de scheduling en máquinas, los cuales se formulan a partir de la lógica y la aritmética difusa.

Los tiempos de procesamiento se formulan a partir de p_{ij}, que representa el tiempo de ejecución del trabajo *i* en la máquina *j* , pudiendo éste último omitirse si el trabajo es independiente de la máquina, sobre todo cuando el trabajo se procesa en una sola máquina (ver [52]). En la mayoría de casos, la formulación de problemas de scheduling se asume de manera determinística. En general, los tiempos de duración o de procesamiento en las industrias de manufactura son a menudo imprecisos y la imprecisión en los datos es muy crítica para los procedimientos de scheduling.

Según Fortemps en [23], los números difusos son considerados como un conjunto de posibles distribuciones probabilísticas. McCahon y Lee [43] utilizan números difusos triangulares y trapezoidales para representar los tiempos de procesamiento ($\widetilde{p_{ij}}$), en donde la función de membresía alcanza su máximo valor al llegar al tiempo de procesamiento más seguro.

En [61], Hshein representa los tiempos de procesamiento fuzzy triangular mediante $\widetilde{p_{ij}} = (p_{ij}^L, p_{ij}, p_{ij}^U)$, en caso que $i = 1, ..., n$ y $j = 1, ..., m$.

La función de membresía viene dada por:

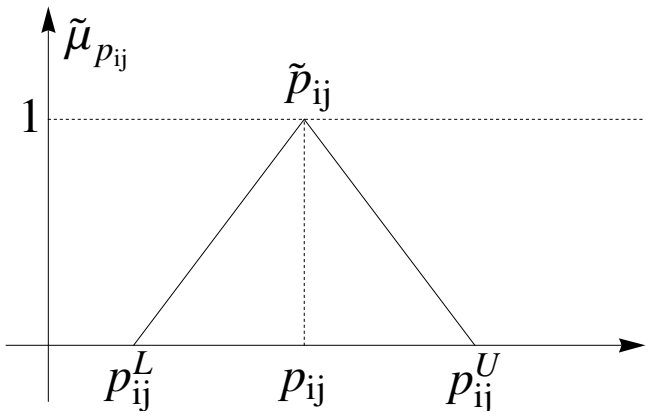

Figura 3.1: Tiempo de procesamiento triangular difuso

$$\mu_{p_{ij}}(\mathbf{P}) = \begin{cases} \dfrac{P - P_{ij}^L}{P_{ij} - P_{ij}^L} & \text{si } P_{ij}^L \leq P \leq P_{ij} \\ \dfrac{P - P_{ij}^U}{P_{ij} - P_{ij}^U} & \text{si } P_{ij} \leq P \leq P_{ij}^U \\ 0 & \text{si } P \leq P_{ij}^L \wedge P_{ij}^U \leq P \end{cases} \tag{3.1}$$

Según McCahon y Lee [43], si se mantiene la difusidad a lo largo del análisis de un problema, es de esperarse que si se formulan los tiempos de procesamiento con números difusos, los resultados sean difusos. Gupta et al. [16] formulan los tiempos de procesamiento en flow shop de forma triangular para resolver problemas con *n* trabajos, 2 máquinas y comparan el tiempo que consumen dos algoritmos diferentes para un mismo entorno.

También se puede considerar una fecha de vencimiento como un parámetro

difuso. Sea C_k el tiempo de culminación, con $k = 1, ..., n$, la función de membresía de la fecha de vencimiento difusa $\widetilde{d_k}$ se define como sigue:

$$\mu_{d_k}(C_k) = \begin{cases} 1 & \text{si } C_k \leq d_k^L \\ 1 - \dfrac{C_k - d_k^L}{d_k^U - d_k^L} & \text{si } d_k^L < C_k < d_k^U \\ 0 & \text{si } d_k^U \leq C_k \end{cases} \tag{3.2}$$

Esta función de membresía permite tomar decisiones y poder formular el grado de satisfacción del tiempo de culminación de cada trabajo. Por ejemplo, si el trabajo se completa antes de la cota d_k^L se tendrá el máximo grado de satisfacción pero, a medida que se supera el tiempo de culminación, esta barrera alcanza un límite superior d_k^U en el cual no se obtiene satisfacción o, como se aprecia en [21] también se pueden obtener niveles de satisfacción negativas. Estas situaciones pueden graficarse como:

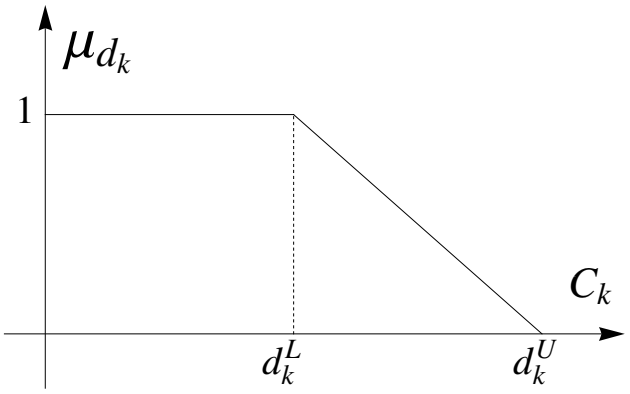

Figura 3.2: Tiempo de culminación difusa

A la diferencia $d_k^U - d_k^L$ se le conoce como rango de tolerancia mínima [32].

Consideremos una secuencia de *n* trabajos, la cual puede representarse como: $J = (J(1), J(2), ..., J(n))$. Entonces podemos definir el tiempo de culminación de la secuencia J como $\widetilde{C_i}(J)$ por cada trabajo. Tomemos $i = J(k)$ y para facilitar la comprensión del concepto analicemos el caso de una máquina. Así, tenemos

$$\widetilde{C_i}(J) = \sum_{j=1}^{k} \widetilde{p}_j.$$

Ahora, conociendo el tiempo de culminación de un conjunto de trabajos, se define la tardanza para cada trabajo *m* en J como el máximo (la función máximo difuso) entre cero y la diferencia (entiéndase aquí la diferencia entre números difusos) entre los tiempos de culminación y la fecha de vencimiento de ese trabajo.

$$\widetilde{T}_m(J) = \widetilde{max}\{0, \widetilde{C}_m - \widetilde{d}_m\}$$

Una relación de precedencia entre trabajos, por lo general, se representa mediante una relación binaria $J_i \prec J_j$, en la cual el trabajo J_j no puede iniciarse sin que antes se haya completado el tiempo de culminación del trabajo J_i. Esta restricción por sí sola no constituye un gran problema, puesto que se puede relajar utilizando restricciones de precedencia difusas por lo tanto basta analizar el nivel de satisfacción en que se ha realizado el trabajo.

En [45] las restricciones de precedencia se dan mediante un orden parcial estricto, el cual es una relación binaria irreflexiva y transitiva del conjunto de trabajos. Así, por ejemplo, vemos que para un conjunto de trabajos $J_i = \{J_1, J_2, ..., J_n\}$ las restricciones de precedencia difusa $J_i \prec J_j$ por cada trabajo

$i = 1, 2, ..., n-1$ bien pueden representarse por el grafo como en la figura 3.3.

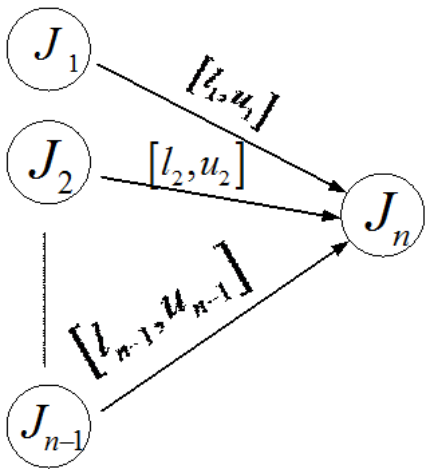

Figura 3.3: Grafo de restricciones de precedencia

Donde los números l_i y u_i representan los límites inferior y superior, respectivamente, para los tiempos de retraso entre cada trabajo J_i y J_n.

Algunas metodologías para los problemas de scheduling se basan en maximizar el mínimo valor del *Agreement Index* o simplemente AI. Como puede observarse en [57] el AI del tiempo de procesamiento de culminación $\widetilde{C}_j$ con respecto a la fecha de vencimiento $\widetilde{D}_j$ es el valor determinado por:

$$AI = \frac{area(\widetilde{C}_j \cap \widetilde{D}_j)}{area(\widetilde{C}_j)}.$$

Gráficamente podemos apreciarlo en la siguiente figura:

Fuzzy scheduling tiene dos clases de aplicaciones: scheduling bajo restricciones flexibles y scheduling bajo información incompleta o imprecisa. Puede considerar en su formulación algunos (o todos) los parámetros difusos. En la medida

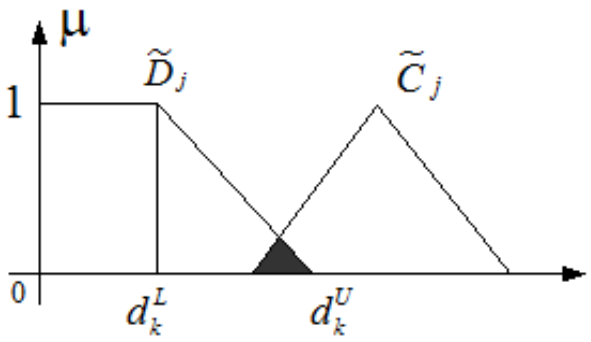

Figura 3.4: Agreement Index

que se incluyan más parámetros, el problema se convierte en NP-duro.

3.1.2. Fuzzy Single Machine Problem

Los problemas de máquinas simples o Single Machine no solo se desarrollan en entornos determinísticos sino también en difusos. Resolver un problema de máquinas simples consiste en desarrollar trabajos en una sola máquina, asumiendo algunas restricciones y justamente en el desarrollo de estas restricciones radica la importancia de este tipo de problemas, pues es posible hacer algunas generalizaciones hacia otros tipos de scheduling. Las investigaciones desarrolladas por Muthusamy et al. [45] tienen como objetivo minimizar el *makespan*, esto es, el máximo tiempo de terminación de un trabajo, sujeto a restricciones, los cuales se generalizan de la manera usual por las relaciones de precedencia. Además incluyen medidas de rendimiento para minimizar el makespan, maximizar el grado de satisfacción de las restricciones de precedencia difusas y maximizar el grado de satisfacción de restricciones de tiempos de retraso.

Kasperski en [35] desarrolla el problema de scheduling fuzzy en forma general

mediante funciones de costo, las cuales asumen condiciones de monotonicidad y regularidad. Ishii & Tada en [31] aproximan el problema de máquinas simples con el objetivo de minimizar el máximo tiempo de retraso en que incurren los trabajos con relaciones de precedencia. Las soluciones se basan en representaciones de grafos para determinar aquellas soluciones óptimas no dominadas. Jadhva & Bajaj en [33] aplican un método sencillo para la toma de decisiones en máquinas simples con tiempos de procesamiento y fechas de vencimiento difusos, el cual debe minimizar un costo de penalidad por cada trabajo que se retrase. Chanas & Kasperski en [12] enfocan el problema de minimizar el máximo tiempo de tardanza con tiempos de procesamiento y fechas de vencimiento ambos difusas, asignando a cada trabajo una función real que contiene el tiempo de procesamiento y la fecha de vencimiento. Posteriormente en [14] además de definir la optimalidad posible y necesaria, se usan parámetros difusos . Éstos terminos se definen a partir del grado de optimalidad posible y necesario para que un determinado schedule sea considerado óptimo (de forma general) y aplicado al problema clásico de una máquina y de tiempos de culminación ponderadas. Dos problemas que incluyen tiempos de procesamiento y fechas de vencimiento difusas con resolución algorítmica pueden encontrarse en [13]. El primer problema minimiza el máximo valor esperado de la tardanza y el segundo minimiza el valor esperado de un máximo de tardanzas difusas, aparentemente similares, pero de diferente complejidad computacional. El primero más sencillo que el segundo.

En las revisiones de single machine es posible ampliar la idea del entorno difuso a múltiples objetivos complementando las metodologías con alguna de las

técnicas metaheurísticas. Podemos citar, por ejemplo el trabajo dado en [44], en donde se explora la tardanza ponderada para single machine en los algoritmos genéticos y en las técnicas para maximizar los mínimos grados de satisfacción y minimizar el número de trabajos tardíos. Otras aproximaciones con técnicas que incluyen múltiples objetivos las encontramos en el trabajo de Duenas & Petrovic [21] en el cual se minimizan el máximo y la tardanza promedio de los trabajos con fechas de vencimiento difusas. Tavakkoli et al. en [59] utilizan métodos de programación lineal difusa para minimizar la tardanza ponderada y el makespan de forma simultánea, nuevamente se consideran grados de satisfacción para tomar decisiones con respectos a los objetivos a optimizar.

3.1.3. Fuzzy Parallel Machine Problem

Un problema de máquinas paralelas o Parallel Machine scheduling consiste en trabajos que llegan a una determinada planta que contiene estaciones y en las cuales se dispone de máquinas, las cuales pueden procesar el trabajo en paralelo. Sobre fuzzy parallel machine, el trabajo de Balin [8] usa tiempos de procesamiento difuso para generar un algoritmo genético que aproxime un modelo de simulación para minimizar el makespan y probar la robustez de la solución ante diversas simulaciones. También Balín en [9] amplía esta idea a máquinas paralelas no idénticas con tiempos de procesamiento difuso. Se consideran no idénticas porque las máquinas poseen velocidades distintas (debido a características tecnológicas) y diferentes tiempos de procesamiento. Alcan & Basligil en [4] utilizan números difusos triangulares para modelar tiempos de procesamiento y aproximan el

mismo problema de máquinas no indénticas con algoritmos genéticos.

También se extiende la idea, en este caso, a los problemas de múltiples objetivos. Por ejemplo, está el estudio de Peng & Liu [49] en donde se formulan modelos con tres objetivos: minimizar la máxima tardanza difusa, el máximo tiempo de culminación difuso y el máximo tiempo difuso en el que las máquinas se encuentran libres. Todo esto se realiza mediante algoritmos híbridos. Asghari & Nezhadali en [6] consideran varios objetivos, entre ellos: minimizar la tardanza total ponderada y el earliness. Además, incluyen el costo de deterioro mínimo que se genera por el deterioro (o penalidades) de las máquinas, como influencia de las finanzas en la producción.

3.1.4. Fuzzy Flow Shop Problem

En un problema de Flow shop los trabajos se procesan en una serie de máquinas en un orden predeterminado, caracterizado por un flujo continuo e ininterrumpido de trabajos en múltiples máquinas en serie. Para Flow Shop scheduling en entornos difusos, Gupta et al. en [28] desarrollan algoritmos heurísticos con el objetivo de minimizar el costo de rentas de máquinas con políticas específicas de rentas que incluyen tiempos de procesamiento difuso representados por números difusos triangulares con 3 máquinas (en [27] se soluciona el problema de dos máquinas). Un modelo matemático para flow shop que, si bien es estocástico, también aprovecha la representación triangular para números difusos, se encuentra en [53]; se formula para optimizar dos criterios significantes: el makespan esperado

y la probabilidad de minimizar el makespan. También se utiliza la representación trapezoidal de números difusos para formular los tiempos de procesos. En [2] se trata de obtener una secuencia de trabajos ordenados para minimizar el máximo tiempo de culminación difusa entre los trabajos. Gonzáles et al. [24] usan métodos híbridos, heurísticos combinados con técnicas de búsqueda local y los comparan con algoritmos genéticos para resolver problemas de job shop con incertidumbre.

3.1.5. Fuzzy Job Shop Problem

Tradicionalmente, para los problemas de Job Shop Scheduling, se han asumido, tres tipos de aproximaciones. Según [18], éstas pueden ser: reglas de prioridad, optimización combinatoria y análisis de restricciones. La idea de trabajar las aproximaciones desde el punto de vista difuso, consiste en dejar la posibilidad de interpretar y manipular la vaguedad en forma numérica, como indica Dubois et al. [18] en su propuesta, se busca mostrar el conjunto de soluciones a un problema como una función de membresía que refleje las preferencias y los niveles de satisfacción en las restricciones. Así, un problema Fuzzy Job Shop no es más que una extensión del problema clásico.

La principal dificultad para resolver el modelo del problema job shop es buscar la ruptura de las restricciones disyuntivas. Para este fin se han desarrollado algunas propuestas que, si bien no resuelven el modelo aproximan la solución con alternativas factibles y controlables. A pesar de que la búsqueda del método para obtener esa ruptura es complicada, por lo general se busca relajar este conjunto

de restricciones.

Una metodología utilizada para tratar el problema de la vaguedad (o difusidad) en la información de los problemas de scheduling, la exponen Kuroda & Wang [38], en donde usan métodos de Branch and bound con algoritmos para obtener soluciones óptimas para criterios de rendimientos dados. Fortemps en [23] utiliza la aproximación por medio de números difusos para modelar técnicas meta-heurísticas que por medio de simulación determinan soluciones únicas y óptimas para el problema de Job Shop Scheduling.

El problema fuzzy de Job Shop Scheduling para problemas del mundo real se formula utilizando los tamaños de lotes en donde se presenta la incertidumbre. Petrovic et al. [50] fuzifican ciertas premisas y conclusiones del problema de job shop como números difusos. Los tamaños de los lotes se ingresan a un algoritmo genético multiobjetivo teniendo en consideración la minimización del tardiness, los tiempos de preparación y el número de trabajos que llegan tarde.

Como puede observarse, otra de las técnicas recientes utilizadas para resolver problemas de Job Shop Fuzzy se basan en el procedimiento de búsqueda iterativa, motivada por el problema de la evolución biológica, los Algoritmos Genéticos Fuzzy, basados en números difusos triangulares, cuyo objetivo es superar los problemas de diversidad poblacional observados en los algoritmos genéticos clásicos. Así lo muestran Carbadillo et al. en [11], cuyo trabajo presentó una propuesta de fuzzificación de los algoritmos genéticos, con el principal objetivo

de superar los problemas de convergencia prematura exhibidos en los algoritmos genéticos tradicionales. También Sakawa & Mori en [55] incorporan los tiempos de procesamiento difuso y fechas de vencimiento en el cual se sitúa el algoritmo genético para maximizar los índices que miden el grado de satisfacción del tiempo de finalización de un determinado trabajo. De forma similar se puede ampliar el concepto de optimizar más de un objetivo. Es así como Fayad & Petrovic inciden en utilizar algoritmos genéticos para resolver problemas fuzzy Job shop scheduling del mundo real a partir de dos criterios: la tardanza promedio y el número de trabajos que llegan. Posteriormente Sakawa & Kubota [54] amplían esta propuesta a tres objetivos no solo maximizando el mínimo índice de satisfacción sino también maximizando el índice promedio de satisfacción y minimizando el máximo tiempo de finalización fuzzy de los trabajos.

Recientemente Sakawa en [57] coloca nuevamente la toma de decisiones en los problemas de scheduling desde un punto de vista de aproximación mediante algoritmos genéticos con parámetros difusos, considerando como elemento importante el índice agregado. Observa, que todo problema de Job shop tiene un Schedule óptimo dentro del conjunto de schedule activos. Esta propuesta la cual mejora el algoritmo de Giffler and Thompson para problemas fuzzy job shop es comparada con el algoritmo Simulated Annealing (SA) el cual incluye un método probabilístico. También, es necesario saber que, para modelar situaciones del mundo real, se deben formular estas situaciones como problemas de múltiples objetivos. En este caso se consideraron tres objetivos en los cuales no solo basta maximizar el mínimo índice agregado (AI), sino también maximizar el índice

promedio agregado y minimizar el máximo tiempo de culminación difusa. Para este nuevo caso se utilizan nuevamente los algoritmos genéticos, tomando en cuenta una representación matricial en la cual los elementos vienen a ser tiempos de procesamiento difuso en cada máquina.

3.2. Algoritmo para $1||\sum \omega_j \widetilde{C}_j$

3.2.1. Variables del problema

Variables de Entrada

- Tiempos de procesamiento (tps). Es una lista de ñëlementos, los cuales están definidos mediante ternas $\{p_1, p_2, p_3\}$ cuyos elementos son las componentes de un número difuso dado de forma triangular.
    - p_1. Indica el mínimo tiempo en que se puede procesar el trabajo.
    - p_2. Indica el valor con mayor grado de incidencia en el procesamiento del trabajo.
    - p_3. Es el máximo tiempo en que se puede procesar un determinado trabajo.
- Ponderaciones (pesos). Es una lista que contiene ñëlementos, los cuales indican un factor de importancia del trabajo en la planificación que se desea elaborar.

Variables de Salida

- Tiempos de culminación C_j. Es una lista de tamaño ñ, el cual está definido mediante ternas $\{C_1, C_2, C_3\}$ cuyos elementos son las componentes de un

número difuso dado de forma triangular. Esta variable indica el tiempo en que un trabajo ha sido procesado por una máquina.

- Schedule S. Contiene el orden de los trabajo que minimiza el máximo tiempo de culminación ponderada $\sum \omega_j \widetilde{C}_j$

3.2.2. Método de solución

Una metodología clásica consiste en el criterio de las proporciones de Smith [39], con la cual es posible encontrar un schedule óptimo ordenando de manera no decreciente las proporciones $\frac{\widetilde{p}_j}{\omega_j}$, que es conocido como el criterio WSPT.

La imprecisión en los tiempos de procesamiento se representó mediante números difusos de tipo triangular $\widetilde{p}_j = (p_{j1}, p_{j2}, p_{j3})$ para cada trabajo $j = \{1, \cdots, n\}$, las proporciones

$$\widetilde{\rho}_j = \frac{\widetilde{p}_j}{\omega_j}$$

Se utilizó el intervalo esperado de cada proporción como medida de comparación para ordenar los trabajos en forma no decreciente. Para cada par de proporciones, la medida en que $\widetilde{\rho}_j$ es mayor que $\widetilde{\rho}_k$ se determinó con la medida de comparación de Jiménez.

Se generó una matriz cuadrada $A = (a_{ij})_{nxn}$ cuyos elementos resultan de la comparación de Jiménez, basado en el intervalo esperado entre todas las

proporciones obtenidas. Es decir

$$a_{ij} = \begin{cases} M\left[\widetilde{\rho}_i, \widetilde{\rho}_j\right] & \text{si } M\left[\widetilde{\rho}_i, \widetilde{\rho}_j\right] > 0,5 \\ 0 & \text{Otro caso} \end{cases} \tag{3.3}$$

Fue necesario generar un indicador para esta matriz que establezca una relación directa con el schedule óptimo. Para ello se obtiene una secuencia de escalares λ_i tal que

$$\lambda_i = \sum_{j=1}^{n} a_{ij} \tag{3.4}$$

Mediante el ordenamiento de estos escalares en un vector se obtiene la secuencia óptima para el problema $1||\sum \omega_j \widetilde{C}_j$

3.2.3. Algoritmo

Algoritmo 1: Algoritmo para problema $1||\sum \omega_j C_j$

Entrada : n: número de trabajos; $\widetilde{p}_j$: Tiempos de procesamiento difusos; ω_j: pesos de los trabajos

Resultados: S: Schedule factible; $\sum_j \widetilde{C}_j \cdot \omega_j$;

`/* Cálculo de las proporciones en los trabajos */`

foreach $i = 1 \cdots n$ **do**

- $\widetilde{\rho}_i = \dfrac{\widetilde{p}_i}{\omega_i}$
- $EI\left[\widetilde{\rho}_i\right] = \left[EI_1^{\widetilde{\rho}_i}, EI_2^{\widetilde{\rho}_i}\right]$

`/* Aplicar la medida de comparación entre las proporciones de los trabajos */`

for $i = 1 \cdots n$ **do**

- **for** $j = 1 \cdots n$ **do**
  - **if** $EI_1^{\widetilde{\rho}_i} - EI_2^{\widetilde{\rho}_j} < 0$ **then**
    - $M\left[\widetilde{\rho}_i, \widetilde{\rho}_i\right] = 0$ **else**
      - **if** $0 \in \left[EI_1^{\widetilde{\rho}_i} - EI_2^{\widetilde{\rho}_j}, EI_2^{\widetilde{\rho}_i} - EI_1^{\widetilde{\rho}_j}\right]$ **then**
        - $M\left[\widetilde{\rho}_i, \widetilde{\rho}_i\right] = \dfrac{EI_2^{\widetilde{\rho}_i} - EI_1^{\widetilde{\rho}_j}}{EI_2^{\widetilde{\rho}_i} - EI_1^{\widetilde{\rho}_j}} - \left(EI_2^{\widetilde{\rho}_1} - EI_2^{\widetilde{\rho}_j}\right)$ **else**
          - $M\left[\widetilde{\rho}_i, \widetilde{\rho}_i\right] = 1$

`/* Generar la matriz de comparaciones y calcular los valores` λ_i `*/`

for $i = 1 \cdots n$ **do**

- $\lambda(i) = 0$ **for** $j = 1 \cdots n$ **do**
  - **if** $M\left[\widetilde{\rho}_i, \widetilde{\rho}_i\right] > 0{,}5$ **then**
    - $A(i,j) = M\left[\widetilde{\rho}_i, \widetilde{\rho}_i\right] = 0$ **else**
      - $A(i,j) = 0$
  - $\lambda(i) = \lambda(i) + A(i,j)$

Ordenar en forma ascendente los elementos del vector $\lambda = \{\lambda(1) \cdots \lambda(n)\}$ y guardarlo en $V = \{V(1) \cdots V(n)\}$

`/* Obtención del schedule óptimo */`

Hacer $V = \{V(1) \cdots V(n)\}$ con $s(k) = pos(v(k))$, $k = 1, \cdots, n$ donde $pos(v(k))$ indica la posición de $v(k)$ en el vector λ

`/* Se calculan los tiempos de culminación */`

for $j = 1, \cdots, n$ **do**

- Calcular $\widetilde{C}_j$

Calcular la función objetivo $\sum \omega_j \widetilde{C}_j$

Imprimir S,$\sum \omega_j \widetilde{C}_j$

3.3. Algoritmo para $1|Prec|\sum \omega_j \widetilde{C}_j$

Utilizando la notación de Graham, el problema a resolver es de tipo $1|Prec|\sum \omega_j \widetilde{C}_j$. En este problema se tiene un conjunto de n trabajos los cuales tienen imprecisión en la duración de sus tiempos de procesamiento, además de cierto factor de importancia para el procesamiento de los trabajos. La importancia es representada mediante un valor $\omega_j \geq 0$ real, mientras que el tiempo de procesamiento es representado utilizando números difusos.

A diferencia del modelo anterior, en este problema se consideran restricciones de precedencia entre trabajos, si el trabajo i precede a j, indica que el inicio de j queda postergada hasta la culminación de i. Esta inclusión en la formulación del problema lo clasifica en un problema de tipo NP - Hard ([26]), es decir aquel algoritmo de ejecución lo resuelve en un orden no polinomial.

El objetivo de este tipo de problemas consiste en encontrar un schedule que minimice $\sum_{i=1}^{n} \omega_j \widetilde{C}_j$

3.3.1. Variables del problema

Variables de Entrada

- Tiempos de procesamiento (tps). Es una lista de n elementos, los cuales están definidos mediante ternas $\{p_1, p_2, p_3\}$ cuyos elementos son las componentes de un número difuso dado de forma triangular.

    - p_1. Indica el mínimo tiempo en que se puede procesar el trabajo.

- p_2. Indica el valor con mayor grado de incidencia en el procesamiento del trabajo.
- p_3. Es el máximo tiempo en que se puede procesar un determinado trabajo.

- Ponderaciones (pesos). Es una lista que contiene n elementos, los cuales indican un factor de importancia del trabajo en la planificación que se desea elaborar.
- Cadenas. Es una lista que contiene el orden de los trabajos formada a partir de la representación mediante grafo de las precedencias. Un problema puede tener diferentes cadenas.

Variables de Salida

- Tiempos de culminación C_j. Es una lista de tamaño n, el cual está definido mediante ternas $\{C_1, C_2, C_3\}$ cuyos elementos son las componentes de un número difuso dado de forma triangular. Esta variable indica el tiempo en que un trabajo ha sido procesado por una máquina.
- Schedule S. Contiene el orden de los trabajo que minimiza el máximo tiempo de culminación ponderada $\sum \omega_j \widetilde{C}_j$

3.3.2. Método de solución

Para implementar este algoritmo, los tiempos de procesamiento en los trabajos fueron formulados mediante números difusos triangulares $\widetilde{p}_j = \{p_{j1}, p_{j2}, p_{j3}\}$ para $j = 1, \cdots, n$ y las cadenas de precedencias a través de un arreglo

$Ch(m) = \{\lambda(m1), \cdots, \lambda(mk)\}$ donde m indica la cantidad de cadenas y k la cantidad de trabajos que intervienen en cada cadena ([51]). Como es de notarse, las cadenas tienen diferente tamaño (cantidad de trabajos) e incluso pueden presentarse trabajos que no pertenezcan a ninguna cadena.

La metodología que se propone para este modelo está basado en el método de la cadena para el caso determinístico de este problema, sin embargo en el sentido difuso el problema es más complejo.

Para cada cadena del grafo de precedencia, utilizando el criterio de ordenamiento usado en el modelo anterior se calcula la menor proporción

$$\widetilde{\rho}_{mk} = \frac{\sum_{r=1}^{k} \widetilde{p}_r}{\sum_{r=1}^{k} \omega_r}$$

. Se selecciona el trabajo que contenga el menor

$$\widetilde{\rho}_{mk}$$

y éstos son incluidos en schedule parcial. Este proceso se repite sucesivamente hasta lograr completar el schedule factible.

3.3.3. Algoritmo

Algoritmo 2: Algoritmo para problema $1|Prec|\sum \omega_j C_j$

Entrada : n: número de trabajos; $\widetilde{p}_j$: Tiempos de procesamiento difusos; ω_j: pesos de los trabajos; Cadena de trabajos $\lambda = (\lambda_1, \cdots, \lambda_m)$

Resultados: S: Schedule factible; $\sum_j \widetilde{C}_j \cdot \omega_j$;

`/* Se genera un vector para el schedule de tamaño n */`

Hacer $S = (s(1), \cdots, s(n))$

for $i = 1 \cdots n$ **do**

    $s(i) = 0$

Hacer $m =$ cantidad de cadenas del diagrama de precedencia.

`/* Comenzar el proceso iterativo */`

Hacer $cont = 1$

for $i = cont, \cdots, n$ **do**

    if $s(i) = 0$ **then**

        for $j = 1, \cdots, m$ **do**

            hacer $t =$Cantidad de trabajos de λ_j

            for $k = 1 \cdots t$ **do**

                $\widetilde{\rho}_{jk} = \frac{\sum_{r=1}^{k} \widetilde{p}_r}{\sum_{r=1}^{k} \omega_r}$

                Escoger una medida de comparación para ordenar estos valores

            Tomar el mínimo de cada proporción en cada cadena y comparar con el resto de mínimos.

            Tomar el (los) trabajos con menor proporción y guardar en $s(i)$ de acuerdo a la cantidad de trabajos.

        else

            $cont = cont + 1$

`/* Se calculan los tiempos de culminación */`

for $j = 1, \cdots, n$ **do**

    Calcular $\widetilde{C}_j$

Calcular la función objetivo $\sum \omega_j \widetilde{C}_j$

Imprimir S,$\sum \omega_j \widetilde{C}_j$

3.4. Modelo de optimización difusa para $1||\sum \omega_j \widetilde{C}_j$

El modelo de optimización difusa propuesto resuelve un problema en single machine donde existen restricciones de precedencia en las tareas y además los tiempos de procesamientos de las tareas en una máquina son números difusos triangulares, este modelo tiene como objetivo minimizar el máximo tiempo de culminación ponderada y encontrar un makespan robusto que queda determinado por el tiempo de culminación de cada trabajo en un schedule óptimo. El modelo fue implementado en Cplex, con resultados óptimos.

3.4.1. Variables

Sea $T = \{1, \cdots, n\}$ el conjunto de tareas a procesar en una máquina.

Variables de Entrada

- Tiempos de procesamiento $\widetilde{P}_j$ (tps). Es una lista de n elementos, los cuales están definidos mediante ternas $\{p_1, p_2, p_3\}$ cuyos elementos son las componentes de un número difuso dado de forma triangular.

- Ponderaciones ω_j(pesos). Es una lista que contiene n elementos, los cuales indican un factor de importancia del trabajo en la planificación que se desea elaborar.

- Precedencia. Es una lista que contiene el orden de los trabajos formada a partir de la representación del grafo de precedencias.

Variables de salida

- Tiempos de inicio difuso de las tareas $\widetilde{x}_j$

- Tiempos de culminación de cada tarea $\widetilde{C}_j$

Donde $\widetilde{P}_j$, $\widetilde{C}_j$ y $\widetilde{x}_j$ son números difusos triangulares, definidos de la siguiente forma

$$\widetilde{x}_j = (x_1(j), x_2(j), x_3(j))$$

$$\widetilde{P}_j = (p_1(j), p_2(j), p_3(j))$$

$$\widetilde{C}_j = (c_1(j), c_2(j), c_3(j))$$

3.4.2. Modelo de optimización

$$\text{mín} \sum \omega_j \widetilde{C}_j \tag{3.5}$$

Sujeto a:

$$\widetilde{x}_i - \widetilde{x}_j \geq \widetilde{P}_j - M \cdot d_{ij} \quad \forall i \neq j \in T \tag{3.6}$$

$$\widetilde{x}_j - \widetilde{x}_i \geq \widetilde{P}_i - M \cdot (1 - d_{ij}) \quad \forall i \neq j \in T \tag{3.7}$$

$$\widetilde{x}_j - \widetilde{x}_i \geq \widetilde{P}_i \quad i \to j, \forall i, j \in T \tag{3.8}$$

$$\widetilde{C}_j = \widetilde{x}_j + \widetilde{P}_j \quad \forall j \in T \tag{3.9}$$

Donde 3.5 es la función objetivo que minimiza el máximo tiempo de culminación ponderada. Las restricciones 3.6 y 3.7 se les conoce como restricciones de no traslape entre tareas, 3.8 es la restricción de precedencia en las tareas y la ecuación 3.9 calcula los tiempos de culminación $\widetilde{C}_j$ de cada tarea $j \in T$. La constante M representa un valor muy grande y la variable binaria d_{ij} viene dada por

$$d_{ij} = \begin{cases} 1 & \text{si la tarea } i \text{ está antes de } j \\ 0 & \text{Otro caso} \end{cases} \tag{3.10}$$

3.5. Método de solución para modelo de optimización en $1|Prec|\sum \omega_j \cdot \widetilde{C}_j$

El método de solución para resolver el modelo de optimización difuso propuesto, se sustenta en el artículo publicado por Nasseri [46], donde utiliza la aritmética difusa en las restricciones y en el uso de una función ranking en la función objetivo para poder minimizar el máximo tiempo de culminación ponderada.

3.5.1. Modelo auxiliar

Se procede a remplazar los números difusos triangulares $\widetilde{x}_i$, $\widetilde{p}_i$, $\widetilde{c}_i$ y la función ranking en el modelo original y agregamos dos restricciones que garantizan el orden de las componentes de los números difusos triangulares cuando se calculan los tiempos de culminación

$$\min \omega_j \left(\frac{c_1(j) + c_2(j) + c_3(j)}{4} \right)$$

Sujeto a:

$$x_1(j) \leq x_2(j) \quad \forall j \in T$$

$$x_2(j) \leq x_3(j) \quad \forall j \in T$$

$$(x_1(i), x_2(i), x_3(i)) - (x_1(j), x_2(j), x_3(j)) \geq (p_1(j), p_2(j), p_3(j)) - M \cdot d_{ij}$$

$$(x_1(j), x_2(j), x_3(j)) - (x_1(i), x_2(i), x_3(i)) \geq (p_1(i), p_2(i), p_3(i)) - M \cdot (1 - d_{ij})$$

$$(x_1(j), x_2(j), x_3(j)) - (x_1(i), x_2(i), x_3(i)) \geq (p_1(i), p_2(i), p_3(i))$$

$$(c_1(j), c_2(j), c_3(j)) = (x_1(j), x_2(j), x_3(j)) + (p_1(j), p_2(j), p_3(j))$$

Aplicando la suma y resta difusa en las restricciones se obtiene

$$\text{mín}\, \omega_j \left(\frac{c_1(j) + c_2(j) + c_3(j)}{4} \right)$$

Sujeto a:

$$x_1(j) \leq x_2(j) \quad \forall j \in T$$

$$x_2(j) \leq x_3(j) \quad \forall j \in T$$

$$(x_1(i) - x_3(j), x_2(i) - x_2(j), x_3(i) - x_1(j)) \geq (p_1(j) - M \cdot d_{ij}, p_2(j) M \cdot d_{ij}, p_3(j) M \cdot d_{ij})$$

$$(x_1(j) - x_3(i), x_2(j) - x_2(i), x_3(j) - x_1(i)) \geq (p_1(i) - M \cdot (1 - d_{ij}), p_2(j) M \cdot (1 - d_{ij}), p_3(j) M \cdot (1 - d$$

$$(x_1(j) - x_3(i), x_2(j) - x_2(i), x_3(j) - x_1(i)) \geq (p_1(i), p_2(i), p_3(i))$$

$$(c_1(j), c_2(j), c_3(j)) = (x_1(j) + p_1(j), x_2(j) + p_2(j), x_3(j) + p_3(j))$$

simplificando las operaciones aritméticas obtenemos el modelo auxiliar

$$\text{mín}\, \omega_j \left(\frac{c_1(j) + c_2(j) + c_3(j)}{4} \right) \tag{3.11}$$

Sujeto a:

$$x_1(j) \leq x_2(j) \quad \forall j \in T$$

$$x_2(j) \leq x_3(j) \quad \forall j \in T$$

$$x_1(i) - x_3(j) \geq p_1(j) - M \cdot d_{ij} \quad \forall i, j \in T$$

$$x_2(i) - x_2(j) \geq p_2(j) - M \cdot d_{ij} \quad \forall i, j \in T$$

$$x_3(i) - x_1(j) \geq p_3(j) - M \cdot d_{ij} \quad \forall i, j \in T$$

$$x_1(j) - x_3(i) \geq p_1(i) - M \cdot (1 - d_{ij}) \quad \forall i \to j \in T$$

$$x_2(j) - x_2(i) \geq p_2(i) - M \cdot (1 - d_{ij}) \quad \forall i \to j \in T$$

$$x_3(j) - x_1(i) \geq p_3(i) - M \cdot (1 - d_{ij}) \quad \forall i \to j \in T$$

$$x_1(j) - x_3(i) \geq p_1(i) \quad \forall j \in T$$

$$x_2(j) - x_2(i) \geq p_2(i) \quad \forall j \in T$$

$$x_3(j) - x_1(i) \geq p_3(i) \quad \forall j \in T$$

$$c_1(j) = x_1(j) + p_1(j) \quad \forall j \in T$$

$$c_2(j) = x_2(j) + p_2(j) \quad \forall j \in T$$

$$c_3(j) = x_3(j) + p_3(j) \quad \forall j \in T$$

Capítulo 4

Aplicación de Modelos

En esta sección se evaluarán los modelos obtenidos mediante la aplicación de la aritmética difusa a los problemas de scheduling en máquinas simples.

4.1. Modelo $1||\omega_j\widetilde{C}_j$

En esta sección aplicaremos el algoritmo a los siguientes conjuntos de trabajos:

Trabajo	Tiempo de procesamiento	Peso
1	$\{2, 4, 7\}$	4
2	$\{9, 13, 16\}$	18
3	$\{5, 8, 11\}$	7
4	$\{8, 11, 15\}$	10
5	$\{10, 15, 20\}$	20
6	$\{7, 11, 14\}$	17
7	$\{9, 14, 17\}$	4
8	$\{7, 11, 15\}$	16
9	$\{10, 14, 17\}$	2
10	$\{6, 9, 12\}$	7

Aplicando la medida de comparación en cada par de trabajos se obtiene la siguiente matriz

$$\begin{pmatrix} 0 & 0{,}932203 & 0 & 0 & 0{,}857143 & 1 & 0 & 0{,}928571 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}689796 & 0 & 0{,}546875 & 0 & 0 \\ 0{,}576271 & 1 & 0 & 0{,}522936 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0{,}564103 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0{,}59375 & 0 & 0 & 0 & 0{,}758065 & 0 & 0{,}625 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0{,}620968 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0{,}711864 & 1 & 0{,}666667 & 0{,}706422 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

y su respectivo λ_j dado por

$$\begin{pmatrix} 3{,}71792 \\ 1{,}23667 \\ 5{,}09921 \\ 4{,}5641 \\ 1{,}97681 \\ 0 \\ 8 \\ 0{,}620968 \\ 9 \\ 6{,}08495 \end{pmatrix}$$

Asociando cada trabajo con su respectivo λ_j para ordenar se obtiene el siguiente arreglo

Trabajo	1	2	3	4	5	6	7	8	9	10
λ_j	3.71	1.23	5.09	4.56	1.97	0	8	0.62	9	6.08

Finalmente el schedule factible se obtiene a partir de la ordenación de la tabla anterior

$$\{6, 8, 2, 5, 1, 4, 3, 10, 7, 9\}$$

Ahora los tiempos de culminación aparecen en la siguiente tabla:

Trabajo	Tiempo de procesamiento	Tiempo de culminación
1	$\{2, 4, 7\}$	$\{35, 54, 72\}$
2	$\{9, 13, 16\}$	$\{23, 35, 45\}$
3	$\{5, 8, 11\}$	$\{48, 73, 98\}$
4	$\{8, 11, 15\}$	$\{43, 65, 87\}$
5	$\{10, 15, 20\}$	$\{33, 50, 65\}$
6	$\{7, 11, 14\}$	$\{7, 11, 14\}$
7	$\{9, 14, 17\}$	$\{63, 96, 127\}$
8	$\{7, 11, 15\}$	$\{14, 22, 29\}$
9	$\{10, 14, 17\}$	$\{73, 110, 144\}$
10	$\{6, 9, 12\}$	$\{54, 82, 110\}$

El makespan obtenido en este caso es:

$$\{73, 110, 144\}$$

cuya gráfica se muestra a continuación

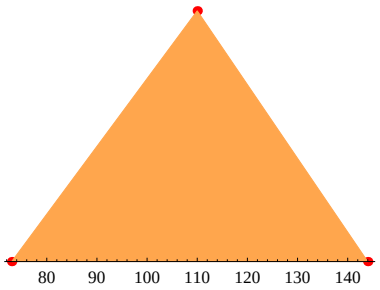

Figura 4.1: Makespan para instancia de Lei

Como puede apreciarse en la figura 4.1 el arreglo de trabajos permiten tener como máximo tiempo de culminación entre las 73 y 144 unidades de tiempo, pero

con un más alto grado de precisión a las 110 unidades de tiempo.

El diagrama de Gantt difuso estaría dado por:

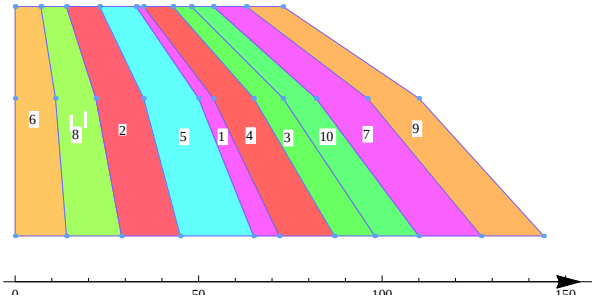

Figura 4.2: Diagrama de Gantt difuso para instancia de Lei

Finalmente la función objetivo obtenida para esta instancia viene dado por el número difuso triangular

$$\{3099, 4724, 6222\}$$

y cuya gráfica se representa en

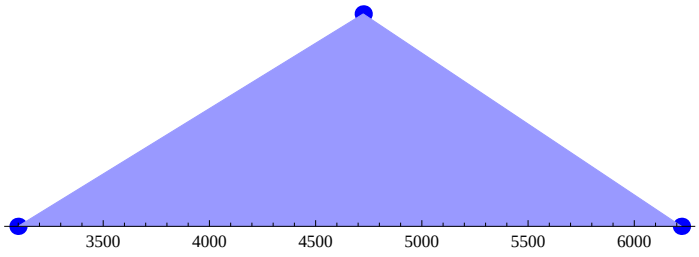

Figura 4.3: Máximo tiempo de culminación ponderada según instancia de Lei

Gráficamente se observa que el más alto grado de posibilidad de la función objetivo se da cuando este valor es igual 4724 unidades, permitiendo una tolerancia entre las 3099 y 6222 unidades de tiempo.

4.2. Modelo $1|Prec|\omega_j\widetilde{C}_j$

Para este caso utilizaremos una instancia propuesta por Alharkan para demostrar la eficiencia del modelo de optimización

Trabajo	Tiempo de procesamiento	Peso
1	$\{1,3,5\}$	6
2	$\{4,6,8\}$	18
3	$\{3,6,7\}$	12
4	$\{3,5,7\}$	8
5	$\{2,4,6\}$	8
6	$\{6,8,10\}$	17
7	$\{8,10,12\}$	18

Luego de resolver el modelo en Cplex se obtuvieron los tiempos de culminación de las tareas 4.2 y el Máximo tiempo de culminación ponderada que se obtiene es: 1941.5 unidades de tiempo.

Trabajo	p_j	ω_j	$\widetilde{x}_j$	$\widetilde{C}_j$
1	$\{1,3,5\}$	6	$\{0,0,0\}$	$\{1,3,5\}$
2	$\{4,6,8\}$	18	$\{1,3,5\}$	$\{5,9,13\}$
3	$\{3,6,7\}$	12	$\{5,9,13\}$	$\{8,15,20\}$
4	$\{3,5,7\}$	8	$\{24,37,48\}$	$\{27,42,45\}$
5	$\{2,4,6\}$	8	$\{8,15,20\}$	$\{10,19,26\}$
6	$\{6,8,10\}$	17	$\{10,19,26\}$	$\{16,27,36\}$
7	$\{8,10,12\}$	18	$\{16,27,36\}$	$\{24,37,48\}$

La secuencia factible bajo este modelo de optimización se da mediante el schedule

$$1, 2, 3, 5, 6, 7, 4$$

y el makespan obtenido fue de $\{27, 42, 45\}$.

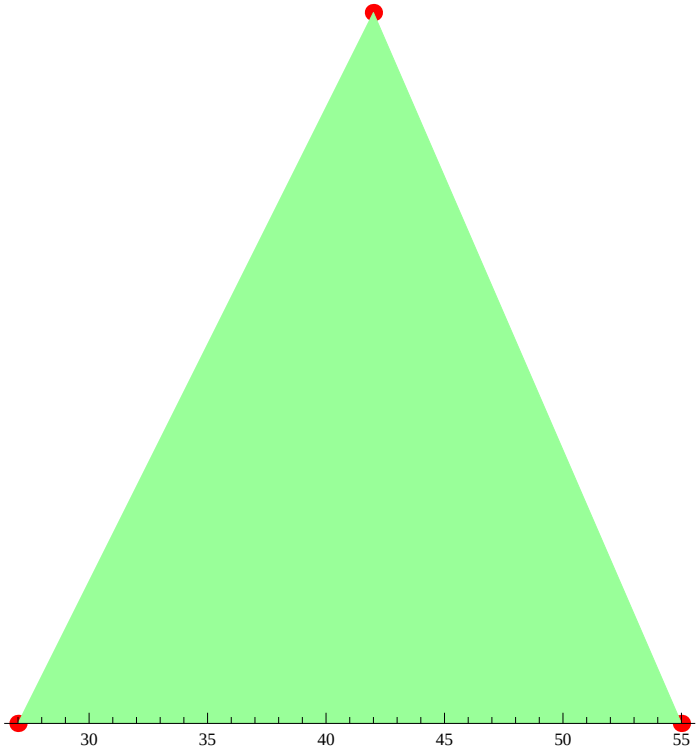

Figura 4.4: Makespan para caso propuesto por Alharkan

Capítulo 5

Conclusiones

5.1. Aportaciones

- Se realizó una revisión sobre el estado del arte en fuzzy scheduling, la cual guió la investigación en el planteamiento de problemas no abordados en la literatura o enfocados mediante técnicas heurísticas o de programación matemática.
- Se realizó una revisión sobre el conocimiento de la aritmética difusa, propiedades de los números así como los diferentes métodos ranking para comparar números difusos, logrando identificar una medida robusta en la comparación y que resume los grados de posibilidad y dominancia en la comparación de números difusos, esta medida corresponde a la comparación por el intervalo esperado.
- Se logró proponer métodos de solución (algoritmos) para la resolución de problemas en máquinas simples considerando principios de extensión de

scheduling clásico hacia entornos difusos.

- Se logró proponer una versión nueva del modelo clásico para problemas de scheduling en máquinas simples y con restricciones de precedencia, el cual considera variables de salida difusas. Este modelo fue probado y programado en CPLEX.

- Los resultados obtenidos mostraron que el procedimiento en la búsqueda mediante la extensión de los criterios clásicos a entornos difusos fue eficiente, esto debido a la satisfacción de las condiciones sobre el cual circunscriben los problemas de scheduling.

5.2. Líneas abiertas

Como resultado de la investigación, se abren posibilidades para futuras investigaciones:

- Desarrollar modelos de optimización difusa para problemas de scheduling considerando la difusidad en todos los elementos intervinientes.

- Extender los modelos propuestos a entornos de máquinas paralelas que contemplen además fechas de vencimientos difusas.

- Explorar técnicas metaheurísticas para la resolución de los problemas abordados en esta investigación con el fin de comparar y establecer mejoras en la búsqueda de soluciones.

5.3. Publicaciones asociadas

- **Un algoritmo eficiente para problemas en single machine con tiempos de procesamiento difusos.**

  E. Lazo, F. Gutiérrez, E. Vergara.

  Revista Cubana de Ciencias Informáticas. Vol.10 no.4 La Habana oct.-dic. 2016. Indexada en Scielo.

  Flabio Gutierrez, es Magister en Matemática Aplicada - Universidad Nacional de Piura (Perú), Magister en Ciencias de la computación por la Universidad de Cantabria (España). Docente del Dpto. de Matemática - Universidad Nacional de Piura. Investigador en Inteligencia Artificial (Sistemas inteligentes, Lógica difusa, Optimización Difusa, Redes Neuronales, Algoritmos Genéticos, etc)..

  Edmundo Vergara, es Dr. en Ciencias Matemáticas - Universidad de Granada (ESPAÑA), profesor principal del Dpto. de Matemática - Universidad Nacional de Trujillo (PERU), Investigador en Optimización y Soft Computing . Miembro de la Sociedad Peruana de Matemática Aplicada y Computacional (SPMAC), Colegio de Matemáticos del Perú.

APÉNDICE A

Benchmark para scheduling

A.1. Benchmark para scheduling

En esta revisión se analizan los casos de estudios propuestos por Taillard en el artículo "Benchmarks for basic scheduling problems"[58], el cual propone un método de asignación aleatoria para generar problemas interesantes en entornos de Flow Shop, Open Shop y Job Shop, cuya función objetivo básico consiste en la minimización del makespan. Este recurso se encuentra disponible en el siguiente link `http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html`.

Para generar las instancias, Taillard considera 3 pasos importantes: La generación de problemas interesantes, el número aleatorio inicial y los problemas específicos. Con respecto al último paso mencionado, los casos de estudios propuestos han sido generados a partir de una clasificación de acuerdo a la complejidad computacional de las soluciones a problemas de flow shop, job shop

y open shop.

A.1.1. Descripción del benchmark

El benchmark propuesto por Taillard constituye una herramienta bastante interesante, porque no solamente presenta casos de estudios bastante diversos y con un método de búsqueda aleatorio eficiente, sino que en cada caso se presentan los mejores resultados obtenidos y su correspondiente autor.

El problema de flow shop puede ser formulado como sigue: considere un grupo de n trabajos $J = \{1, 2, \ldots, n\}$ que debe ser procesado en un conjunto de m máquinas en el orden dado por el índice al que se le fija la máquina. Los mejores ejemplos que encajan en este tipo de problemas son los relacionados a las líneas de ensamblaje. Cada trabajo j, $j \in J$ consiste de una secuencia de m operaciones, las cuales corresponden al procesamiento del trabajo j en la máquina i durante un tiempo de procesamiento $p_{ij} \geq 0$ ininterrumpido [47]. Un schedule para estos problemas viene dado por la permutación π del conjunto de trabajos J. Si se consideran todas las permutaciones posibles para este conjunto de trabajos $\pi \in \Pi$ es posible teóricamente determinar un schedule que minimice el máximo tiempo de culminación (makespan).

Taillard [58], genera los tiempos de procesamiento p_{ij} a partir de un función de distribución uniforme $U\,[1; 99]$. Además se consideran las siguientes variables:

- b_i : representa la menor cantidad de tiempo antes que la máquina i empiece a trabajar.

- a_i : representa la cantidad mínima de tiempo que permanece inactivo de empezar a trabajar para terminar las operaciones.
- T_i es el tiempo total de procesamiento, donde

El benchmark presenta las instancias para problemas de 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20 y 500x20 trabajos por máquina. Las instancias incluyen cuatro elementos necesarios: Semilla inicial (es el valor inicial del generador de semillas aleatorias), LB , UB y los tiempos de procesamiento para cada trabajo. A continuación se presenta un ejemplo para una instancia de flow shop schedulling obtenida del link anterior:

- Número de trabajos: 20
- Número de máquinas: 5
- Semilla inicial: 873654221
- LB: 1239
- UB: 1278

los datos para esta instancia se presentan en el siguiente cuadro:

En `http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/flowshop.dir/best_lb_up.txt` se presenta el mejor valor obtenido para el makespan etiquetado en función al autor que generó dicho aporte, según las referencias bibliográficas que ahí se incluyen.

Thaillard también presenta una base de datos para problemas de job shop schedulling con el objetivo de minimizar el makespan. Los tiempos de procesa-

Tabla A.1: Instancia de Thaillard para problema de flow shop schedulling, [58]

Máquina	Tiempos de procesamiento de cada trabajo
1	54 83 15 71 77 36 53 38 27 87 76 91 14 29 12 77 32 87 68 94
2	79 3 11 99 56 70 99 60 5 56 3 61 73 75 47 14 21 86 5 77
3	16 89 49 15 89 45 60 23 57 64 7 1 63 41 63 47 26 75 77 40
4	66 58 31 68 78 91 13 59 49 85 85 9 39 41 56 40 54 77 51 31
5	58 56 20 85 53 35 53 41 69 13 86 72 8 49 47 87 58 18 68 28

miento d_{ij} son obtenidos de manera similar a los problema de flow shop, por medio de una distribución uniforme $U\,[1; 99]$.

Los problemas de tipo Job shop están caracterizados por su alto grado de complejidad, computacionalmente se encuentran en el orden de los NP-Hard, es decir, los algoritmos que lo desarrollan no pueden ser ejecutados en tiempos polinomiales.

En cuanto a job shop se han generado instancias para problemas de 15x15, 20x15, 20x20, 30x15, 30x20, 50x15, 50x20 y 100x20 trabajos por máquina.

Las instancias para estos problemas presentan las mismas características como las que se refieren a flow shop. Pero de acuerdo a la naturaleza del problema de job shop, las operaciones se pueden desarrollar en distintas máquinas y no necesariamente en el mismo orden, por ello es que se incluye una matriz

que contiene las secuencias de los trabajos que deben desarrollar las distintas máquinas. La siguiente tabla presenta una instancia para el caso de 15 máquinas y 15 trabajos.

Instancia para job shop de 15x15, [58]

Tiempos de procesamiento														
94	66	10	53	26	15	65	82	10	27	93	92	96	70	83
74	31	88	51	57	78	8	7	91	79	18	51	18	99	33
4	82	40	86	50	54	21	6	54	68	82	20	39	35	68
73	23	30	30	53	94	58	93	32	91	30	56	27	92	9
78	23	21	60	36	29	95	99	79	76	93	42	52	42	96
29	61	88	70	16	31	65	83	78	26	50	87	62	14	30
18	75	20	4	91	68	19	54	85	73	43	24	37	87	66
32	52	9	49	61	35	99	62	6	62	7	80	3	57	7
85	30	96	91	13	87	82	83	78	56	85	8	66	88	15
5	59	30	60	41	17	66	89	78	88	69	45	82	6	13
90	27	1	8	91	80	89	49	32	28	90	93	6	35	73
47	43	75	8	51	3	84	34	28	60	69	45	67	58	87
65	62	97	20	31	33	33	77	50	80	48	90	75	96	44
28	21	51	75	17	89	59	56	63	18	17	30	16	7	35
57	16	42	34	37	26	68	73	5	8	12	87	83	20	97

Secuencia de trabajos por máquina														
7	13	5	8	4	3	11	12	9	15	10	14	6	1	2
5	6	8	15	14	9	12	10	7	11	1	4	13	2	3
2	9	10	13	7	12	14	6	1	3	8	11	5	4	15
6	3	10	7	11	1	14	5	8	15	12	9	13	2	4
8	9	7	11	5	10	3	15	13	6	2	14	12	1	4
6	4	13	14	12	5	15	8	3	2	11	1	10	7	9
13	4	8	9	15	7	2	12	5	6	3	11	1	14	10
12	6	1	8	13	14	15	2	3	9	5	4	10	7	11
11	12	7	15	1	2	3	6	13	5	9	8	10	14	4
7	12	10	3	9	1	14	4	11	8	2	13	15	5	6
5	8	14	1	6	13	7	9	15	11	4	2	12	10	3
3	15	1	13	7	11	8	6	9	10	14	2	4	12	5
6	9	11	3	4	7	10	1	14	5	2	12	13	8	15
9	15	5	14	6	7	10	2	13	8	12	11	4	3	1
11	9	13	7	5	2	14	15	12	1	8	4	3	10	6

A parte de presentar las instancias, Taillard incluye las mejores soluciones obtenidas según diferentes autores en diversas etiquetas. Al respecto es importante precisar que no todas éstas poseen una solución óptima, dando origen así a las conocidas instancias duras de Taillard.

APÉNDICE B

Benchmark para scheduling

B.1. Introducción

La literatura sobre Benchmarks para problemas de fuzzy schedulling, muestran experimentos numéricos propuestos por diversos autores a partir de diferentes técnicas de obtención de valores enteros aleatorios.

Los casos de estudios se involucran directamente con el tipo de restricciones que abordan. Al respecto Abdullah y Abdolrazzagh-Nezhad [1], explican 3 tipos de restricciones para FJSSP: precedencia, capacidad, fechas de lanzamiento y vencimiento. La mayoría de estas restricciones se relajan para simular la imprecisión.

La mayoría de los benchmarks propuestos para FJSSP, son generados mediante técnicas aleatorias similares a las utilizadas para generar benchmarks determinísticos. Abdullah [1], clasifica estos casos de estudio en dos grupos: base

de datos fuzzy y base de datos fuzificados. Estas bases de datos implementados o generados han servido para medir el rendimiento de modelos y métodos propuestos para un mismo problema.

B.2. Benchmarks para FJSSP

B.2.1. Benchmark SM

Es uno de los benchmark difusos muy utilizado en la literatura para problemas de fuzzy job shop . Fue formulado por Sakawa y Mori [55], en la que proponen una base de datos (aunque no se menciona el método en que fueron implementados) para FJSSP, consideran a la imprecisión de los tiempos de procesamiento como números difusos triangulares, el objetivo es maximizar el Agreement Index para problemas con fechas de vencimiento, también difusos.

La base contiene instancias para problemas de 6x6 y 10x10 (máquinas y trabajos) en la cual los algoritmos son comparados con otras simulaciones (SA) implementadas. La siguiente tabla muestra una instancia para estos problemas.

Tiempos de procesamiento difuso en máquinas						
Job 1	$1(5,6,13)$	$5(3,4,5)$	$2(1,2,3)$	$6(3,4,5)$	$4(2,3,4)$	$3(2,3,4)$
Job 2	$1(3,4,5)$	$2(2,4,5)$	$3(1,3,5)$	$6(4,5,6)$	$4(5,6,7)$	$5(6,7,8)$
Job 3	$3(1,2,3)$	$6(5,6,7)$	$5(4,5,6)$	$4(3,4,5)$	$2(1,2,3)$	$1(1,2,3)$
Job 4	$6(2,3,4)$	$5(1,2,3)$	$4(2,3,4)$	$2(2,3,5)$	$1(3,4,6)$	$3(3,4,5)$
Job 5	$6(3,4,5)$	$5(2,3,4)$	$4(1,2,3)$	$3(2,3,4)$	$2(4,5,6)$	$1(2,3,4)$
Job 6	$5(6,7,8)$	$6(4,5,6)$	$1(2,3,4)$	$2(3,4,5)$	$3(2,3,4)$	$4(1,2,3)$

	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6
Fuzzy duedate	$30,40$	$35,40$	$20,28$	$32,40$	$30,35$	$40,45$

Sakawa y Kubota [54], proponen una base de datos para problemas de 6x6 y 10x10 máquinas y trabajos con fechas de vencimiento difusas para probar el rendimiento de un nuevo algoritmo para optimizar simultáneamente varias funciones objetivos.

B.2.2. Benchmark Lei

Lei propone instancias para el problema de job shop con restricciones de disponibilidad [41]. Las instancias abarcan problemas de 15 máquinas y 10 trabajos, una de ellas se muestra en la siguiente figura.

Table A1
The processing data of Problem 5.

Job	Processing time and processing sequences									
Job 1	{2, 4, 7}3	{9, 13, 16}4	{5, 8, 11}6	{8, 11, 15}10	{10, 15, 20}5	{7, 11, 14}7	{9, 14, 17}1	{7, 11, 15}9	{10, 14, 17}2	{6, 9, 12}8
Job 2	{8, 11, 15}4	{7, 10, 12}3	{4, 6, 8}1	{11, 15, 20}2	{12, 17, 21}10	{9, 11, 15}9	{8, 12, 16}7	{5, 6, 9}6	{8, 10, 13}5	{7, 11, 15}8
Job 3	{7, 9, 12}2	{6, 7, 9}1	{9, 13, 17}4	{10, 15, 20}5	{5, 8, 12}7	{11, 17, 19}10	{8, 12, 16}9	{7, 9, 13}6	{8, 13, 17}3	{9, 14, 18}8
Job 4	{10, 14, 18}5	{11, 15, 21}3	{9, 13, 17}9	{8, 12, 16}6	{7, 11, 12}4	{11, 15, 19}8	{10, 14, 18}2	{8, 13, 17}7	{9, 14, 18}10	{3, 5, 8}1
Job 5	{7, 10, 13}9	{7, 11, 15}10	{8, 12, 15}3	{9, 13, 16}5	{10, 14, 17}4	{9, 12, 16}1	{10, 15, 17}8	{10, 13, 15}7	{11, 14, 17}2	{9, 12, 16}6
Job 6	{11, 15, 21}9	{9, 15, 18}8	{8, 12, 16}7	{10, 13, 16}10	{7, 11, 14}3	{9, 13, 17}2	{8, 12, 15}6	{10, 14, 17}5	{8, 12, 18}1	{12, 17, 20}4
Job 7	{6, 9, 12}5	{7, 10, 13}6	{8, 11, 15}10	{9, 12, 16}4	{7, 11, 14}1	{8, 10, 14}9	{10, 14, 16}7	{7, 11, 15}8	{7, 10, 11}3	{5, 8, 11}2
Job 8	{9, 12, 16}6	{7, 10, 13}5	{8, 11, 14}3	{6, 9, 13}7	{4, 7, 9}2	{9, 13, 17}8	{8, 10, 13}1	{7, 8, 11}4	{8, 9, 12}10	{6, 8, 10}9
Job 9	{5, 8, 11}2	{7, 12, 14}6	{8, 10, 13}1	{6, 7, 8}4	{4, 5, 8}3	{7, 9, 11}8	{8, 11, 13}9	{9, 10, 14}7	{7, 9, 12}10	{6, 8, 12}5
Job 10	{4, 5, 8}3	{7, 10, 12}6	{8, 12, 16}7	{5, 8, 11}10	{3, 5, 8}2	{4, 7, 9}4	{6, 9, 12}9	{7, 10, 12}1	{5, 7, 9}8	{10, 14, 17}5
Job 11	{7, 8, 11}2	{8, 9, 12}5	{3, 5, 8}1	{5, 7, 10}3	{6, 9, 11}10	{8, 10, 13}9	{7, 10, 11}6	{4, 5, 7}4	{7, 11, 12}8	{9, 13, 17}7
Job 12	{6, 8, 11}6	{4, 7, 10}10	{5, 6, 9}1	{6, 9, 12}5	{5, 8, 10}7	{3, 5, 9}4	{4, 6, 9}3	{5, 8, 12}2	{6, 9, 12}9	{4, 7, 10}8
Job 13	{3, 5, 9}6	{7, 10, 12}10	{5, 7, 9}9	{6, 9, 11}8	{4, 6, 9}5	{8, 10, 13}7	{9, 11, 15}4	{7, 11, 13}1	{5, 8, 9}2	{7, 8, 10}3
Job 14	{5, 8, 11}2	{5, 7, 8}9	{4, 5, 8}1	{7, 11, 14}3	{6, 9, 12}10	{5, 9, 10}4	{4, 5, 6}6	{8, 11, 14}7	{6, 9, 13}5	{5, 8, 11}8
Job 15	{8, 11, 15}5	{7, 10, 12}4	{6, 9, 10}7	{5, 9, 10}6	{7, 9, 12}3	{8, 10, 13}9	{4, 6, 9}2	{4, 7, 10}10	{6, 7, 11}8	{3, 5, 8}1

Como puede apreciarse, los tiempos de procesamiento vienen dados en forma de numeros difusos triangulares, por ejemplo para el trabajo 1, $\{2; 4; 7\}3$ es un tiempo de procesamiento impreciso representado por el numero difuso triangular $\{2, 4, 7\}$, que es lo que necesita la máquina 3 para concluir la operación. Estas instancias han sido implementadas como medida de comparación para lo propuesto por Sakawa y Mori. También puede encontrarse en el artículo cuadros con los mejores rendimientos obtenidos respecto a las instancias implementadas.

B.2.3. Benchmark Iscop

El grupo iScOp - Intelligent Scheduling and Optimization Research Group de la universidad de Oviedo - España poseen en su repositorio digital un conjunto de Benchmarks para problemas de Fuzzy Schedulling, los cuales se encuentran organizados de una manera didáctica que permite facilitar la ubicación del contexto del problema y su relación con el autor de la instancia determinística (no fuzzificada), de ellos se puede mencionar algunos:

- Benchmark para problemas de shcedulling con operadores humanos calificados.
- Benchmark para problemas de single machine con secuencias dependientes

del tiempo de preparación y minimización de la tardanza total promedio.

- Benchmark para Fuzzy Job Shop.
- Benchmark para Fuzzy Flexible Job Shop

Información completa sobre estos Benchmarks pude obtenerse en el link `http://di002.edv.uniovi.es/iscop/index.php?option=com_remository&Itemid=14&func=selec`

Recientemente, miembros del grupo iScOp, como Palacios et al. presentan un artículo titulado "Benchmark for Fuzzy Job Shop"[48], en donde se sistematizan los resultados obtenidos hasta este momento en cuanto a la fuzzificación de instancias clásicas para problemas de Job Shop Schedulling. Presentan por ejemplo, un conjunto de instancias generados por la fuzzificación de los casos de estudios más duros propuestos por Taillard [58], y otros obtenidos de la fuzzificación de Fisher & Thompson [22], Lawrence [40], Adams et al. [3], Benchmark originales [3], así mismo se proponen otras versiones generadas por diversos autores para instancias selectas que aún no logran ser resueltas en las cuales los métodos para resolver no logran ubicarse por debajo del lower bound (LB).

Las instancias fuzzificadas contienen la siguiente información:

- Número de trabajos
- Número de recursos (número de máquinas)
- Matriz de secuencia $m\,[i,j]$ la máquina j toma la tarea de i.

- Duraciones difusas de los tiempos de procesamiento del trabajo i en la máquina j, expresada por la terna (d_1, d_2, d_3).

- Fecha de vencimiento difusa expresada en el par (d_1, d_2).

En el link anterior también se puede ubicar los resultados obtenidos a partir de las metodologías utilizadas para implementar las instancias. Por ejemplo, para generar las instancias fuzzificadas de Taillard, han sido consideradas las instancias del 21 al 50 de la versión clásica, los mejores resultados son presentados en una tabla que contiene los siguientes elementos:

- Instancias : consideradas las instancias más duras de Taillard fuzzificadas desde las instancias 21 a 50.

- Tamaño: muestra el tamaño de las tareas y máquinas, en este caso son consideras las de 20x20 y 30x20.

- LB: contiene el límite inferior utilizado como medida de comparación.

- Method: contiene la metodología propuesta para, se mencionan 2 en especial, GRASP y MA.

- El mejor makespan: contiene el makespan mínimo hallado dado en forma de número difuso triangular, así como el valor esperado del makespan.

- Makespan promedio

- Error relativo: obtenido a través de una medida de comparación para el mejor y el promedio del makespan.

- Tiempo de ejecución de la metodología dado en segundos.

La medida para determinar el error relativo está dado por [48]:

$$RE = \frac{|E\left[C_{max}\right] - LB|}{LB} \tag{B.1}$$

Con esta misma presentación también se muestran resultados para instancias fuzzificadas de:

- Fisher & Thompson [22], con instancias de tamaño 6x6, 10x10 y 20x5 con tiempos de ejecución muy bajos.
- Lawrence [40], con instancias de 10x5, 10x10, 15x5, 15x10, 15x5 y 20x5.
- Becnhmark originales para job shop [5], con instancias para problemas de 10x10.

APÉNDICE C

Programación

C.1. Programa en Wolfram Mathematica para $1||\sum \omega_j \widetilde{C}_j$

```
AlgoritmoSingleMachine[tps_, pesos_] :=
Module[{n, proporcion, Intesp, Vectmu, a, b, ConsTrab, Trab, Vect, VectSol,
Sol, r, m},
  n = Length[tps];
(*Paso 1:Expresamos las proporciones en funcion del alfa corte p*)
  proporcion = Integrate[
     Refine[Table[{Min[(tps[[i,1]]+ p*(tps[[i,2]]-tps[[i,1]]))/
     pesos[[i]],
     (tps[[i,3]] - p*(tps[[i, 3]] - tps[[i, 2]]))/pesos[[i]]],
     Max[(tps[[i, 1]] + p(tps[[i, 2]] - tps[[i, 1]]))/
     pesos[[i]], (tps[[i,3]]-p(tps[[i,3]]-tps[[i,2]]))/
     pesos[[i]]]}, {i, n}], 0 <= p <= 1], {p, 0, 1}];
  Print[Grid[Prepend[Table[{i, tps[[i]], pesos[[i]]}, {i, n}],
```

```
    {"Trabajo","Tiempo de procesamiento", "Pesos"}],Frame->All,
    Background->{{Pink},{Cyan}}]];

(*Paso 2: Cargamos los intervalos esperados en Intesp*)
 Intesp = Array[ie, {n, 2}];
  For[i = 1, i <= n, i++,
    For[j = 1, j <= 2, j++,
       ie[i, j] = proporcion[[i, j]]
     ]
   ];

(*Paso 3:Aplicamos la medida de comparacion a los intervalos
obtenidos*)
  Vectmu = Array[mu, {n, n}];
  For[i = 1, i <= n, i++,
   For[j = 1, j <= n, j++,
    a = Intesp[[i, 1]] - Intesp[[j, 2]];
    b = Intesp[[i, 2]] - Intesp[[j, 1]];
    If[b < 0, r = 0, If[a <= 0 <= b, r = b/(b - a) // N, r = 1]];
    If[r > 0.5, mu[i, j] = r, mu[i, j] = 0 ]] ];

(*Cargamos los la constante asociada a cada trabajo trabajo
(suma de los valores de mu)*)
```

```
  ConsTrab = Array[ct, n];
  Trab = Array[T, n];
  Vect = Array[v, n];
  For[i = 1, i <= n, i++,
     ct[i] = Sum[mu[i, k], {k, 1, n}];
     T[i] = i;
     v[i] = {T[i], ct[i]};
   ];

  Print["Se muestra la matriz de comparación con los valores \
asociados a su respectivo trabajo"];
  Print[Vectmu // MatrixForm, ConsTrab // MatrixForm];
  Print["El vector para ordenar es"];
  Print[Vect];
  VectSol = Sort[Vect, #1[[2]] < #2[[2]] &];

(*Tomamos el orden de los trabajos*)
  Sol = Array[s, n];
  For[i = 1, i <= n, i++,
     s[i] = VectSol[[i, 1]];];
  Print["Se presenta el schedule factible"];
  Print[Sol];
```

```
(*Calculamos los tiempos de culminación*)
  Pos = Table[Position[Sol, k][[1, 1]], {k, 1, n}];
  Culm = Array[culm, n];
  For[k = 1, k <= n, k++,
    a1 = Pos[[k]];
    culm[k] = Sum[tps[[Sol[[z]]]], {z, 1, a1}]
   ];
  Print["Los tiempos de culminación serían"];
    Print[Grid[
      Prepend[Table[{i, tps[[i]], culm[i]}, {i, n}], {"Trabajo",
        "Tiempo de procesamiento", "Tiempo de culminacion"}],
        Frame -> All, Background -> {{Pink}, {Cyan}}]];
  Print["El makespan calculado es"];
  Print[Culm[[Sol[[n]]]]];

(*Calculamos el valor objetivo*)
  Print["El valor objetivo es:"];
  Makespan = Sum[pesos[[k]] culm[k], {k, 1, n}];
  Print[ Makespan];

(*Diagrama de Gantt*)
  Print["El diagrama de Gantt difuso:"];
  culmin = Table[culm[Sol[[z]]], {z, 1, n}];
```

```
h = 100;
m = Max[Flatten[culmin]];
culmlista = Prepend[culmin, {0, 0, 0}];
r = Length[culmlista];
culminacion = Array[G, r];
caja = Array[H, n];
For[i = 1, i <= r, i++,
 G[i] = Partition[Riffle[culmlista[[i]],{(3h)/4,h/2,h/8}],2];];
For[i = 1, i <= r - 1, i++,
 H[i] = {G[i][[1]], G[i][[2]], G[i][[3]], G[i + 1][[3]],
   G[i + 1][[2]], G[i + 1][[1]], G[i][[1]]}];
Eje = Graphics[Arrow[{{0, 0}, {m + 10, 0}}]];
Part1 =
 Graphics[
  Table[{Opacity[.62], Hue[RandomReal[]], Polygon[caja[[i]]],
    Hue[0.688], Thickness[0.0015], Line[caja[[i]]], Hue[0.6],
    Point[caja[[i]]]}, {i, 1, n}], PlotRange -> All];
Show[Eje, Part1, Axes -> {True, False},
 AxesLabel -> Tiempo de Culminacion]
]
```

C.2. Programa en Wolfram Mathematica para $1|Prec|\sum \omega_j \widetilde{C}_j$

Subprograma para ubicación y selección de cadenas.

```
SupChain[Lista_List, elementos_List] := Module[{a, residuo, n, p},
  n = Length[elementos];
  residuo = Lista;
  For[p = 1, p <= n, p++,
   a = Delete[residuo, Position[residuo, elementos[[p]]]];
   residuo = a;
   ]; a
  ]
```

Subprograma para ubicación de elementos en cadenas

```
SupDat[Lista_List, elementos_List] :=
 Module[{n, m, a, residuo, t, trab, z, w, k, u},
  n = Length[Lista];
  m = Length[elementos];
  trab = Array[z, n];
  residuo = Lista;
  For[t = 1, t <= n, t++, z[t] = Datos[[t, 1]]];
  For[w = 1, w <= m, w++,
   k = Position[trab, elementos[[w]]];
   u = Delete[trab, k];
```

```
    trab = u;
    a = Delete[residuo, k];
    residuo = a;
    ];
   a
   ]

UbicDat[Lista_List, elementos_List] :=
 Module[{s, resp, n, r, m, a, residuo, t, trab, z, w, k, u},
  n = Length[Lista];
  trab = Array[z, n];
  resp = Array[r, Length[elementos]];
  residuo = Lista;
  For[t = 1, t <= n, t++, z[t] = residuo[[t, 1]]];
  For[s = 1, s <= Length[elementos], s++,
   r[s] = Flatten[Position[trab, elementos[[s]]]] [[1]]];
  resp
  ]
```

En este subprograma se presenta el método de la cadena.

```
MetodoCadena[Datos_List, c_List] :=
 Module[{D1, D2, n, m, sched, Cd, cont, aux, R, r, S, T, t, z, h, a,
   b, l, d, dd, caux},
  n = Length[Datos];
  sched = Array[Cd, n];
```

```
D1 = Datos;
D2 = c;
m = Length[D2];
aux = Array[R, m];
For[r = 1, r <= n, r++, Cd[r] = 0];
For[cont = 1, cont <= n, cont++,
 If[Cd[cont] == 0, Print[cont];
   For[i = 1, i <= Length[D2], i++, t = Length[D2[[i]]];
    If[t == 0, D2 = Extract[D2, D2[[i]]],
     For[j = 1, j <= t, j++, S = Array[T, t];
      l = Sum[D1[[UbicDat[D1, {D2[[i, k]]}], 2]], {k, 1, j}]/
       Sum[D1[[UbicDat[D1, {D2[[i, k]]}], 3]], {k, 1, j}];
      dd = l[[1]];
      T[j] = {dd, Table[D2[[i, k]], {k, 1, j}]};
      ];
     (*Tomar los mínimos de cada cadena*)
     R[i] = Sort[S, #1[[1]] < #2[[1]] &][[1]];
     Print[S];
     z = Sort[aux, #1[[1]] < #2[[1]] &][[1]];
     ];
    ];
   Print[z];
   k = 1;
```

```
      caux = cont;
      While[k <= Length[z[[2]]],
       d = z[[2, k]];
       Cd[caux] = d;
       k++;
       caux++;
       ];
      Print[sched];
      ];
    ]
  ]
```

C.3. Programa en CPLEX para $1||\sum \omega_j \widetilde{C}_j$

El siguiente modelo de optimización difusa resuelve el problema de minimizar el máximo tiempo de culminación ponderada de un conjunto de tareas para un problema de single machine con tiempos de procesamientos difusos y precedencia en las tareas , el modelo fue implementado en Cplex, obteniendo resultados óptimos.

```
//  Modelo de optimización difuso para problemas de single machine
con tiempos de procesamientos difusos y precedencia en las tareas.
 //
 //   Se enumeran las tareas
range tarea =1..7;
```

```
// variables de  entrada :Tiempos de procesamientos difusos
int P1[tarea] = [1,4,3,3,2,6,8];
int P2[tarea] = [3,6,6,5,4,8,10];
int P3[tarea] = [5,8,7,7,6,10,12];

 //  Variable de entrada : Se ingresan los pesos de cada tarea.
int W[tarea] = [6,18,12,8,8,17,18];
// Se ingresa las precedencias de lastareas.
tupleprecedencia{
int a; int b;};
{precedencia} precedencias = {<1, 2>, <2, 3>, <3, 4>,<5,6>,<6,7>};
// Se ingresa las no conexiones entre las tareas
tuplenoprecedencia{
int c; int d;};
  {noprecedencia} noprec = {<1,5>,<1,6>,<1,7>,<2,5>,<2,6>,<2,7>,
<3,5>,<3,6>,<3,7>,<4,5>,<4,6>,<4,7>};
int M=10000;

// variable de decision

dvarint+ x1[tarea]in 0..100;
dvarint+ x2[tarea]in 0..100;
dvarint+ x3[tarea]in 0..100;
```

```
dvarint+ c1[tarea]in 0..100;
dvarint+ c2[tarea]in 0..100;
dvarint+ c3[tarea]in 0..100;

dvarboolean Z[noprec];

 // funcion objetivo
minimize sum ( j in tarea)(c1[j]+2*c2[j]+c3[j])*W[j]/4;
// restricciones
subjectto
  {
forall(k in tarea){
x1[k]<=x2[k];
x2[k]<=x3[k];
 }
forall(t in noprec){
x1[t.c]>= x1[t.d]+ P1[t.d]-M*Z[t];
x2[t.c]>= x2[t.d]+ P2[t.d]-M*Z[t];
x3[t.c]>= x3[t.d]+ P3[t.d]-M*Z[t];
x1[t.d]>= x1[t.c]+ P1[t.c]-M*(1-Z[t]);
x2[t.d]>= x2[t.c]+ P2[t.c]-M*(1-Z[t]);
x3[t.d]>= x3[t.c]+ P3[t.c]-M*(1-Z[t]);
```

```
}
forall(t in precedencias){
x1[t.b] >=  x1[t.a]+P1[t.a];
x2[t.b] >=  x2[t.a]+P2[t.a];
x3[t.b] >=  x3[t.a]+P3[t.a];
}
  // Datos de salida: tiempos de culminación difusos,
  tiempo de inicios de las tareas.
forall(j in tarea){
c1[j]==x1[j]+P1[j];
c2[j]==x2[j]+P2[j];
c3[j]==x3[j]+P3[j];
}
    };
```

Referencias Bibliográficas

[1] S. Abdullah and M. Abdolrazzagh-Nezhad. Fuzzy job-shop scheduling problems: A review. *Information Sciences*, 278:380–407, 2014.

[2] M. Abolhasani Ashkezari. Shahsavari pour n. and mohammadi andargoli, h.(2012), an ant colony system for solving fuzzy flow shop scheduling problem. *International Journal of Engineering and Technology*, 1(2):44–57.

[3] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401, 1988.

[4] P. Alcan and H. BaşLıGil. A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem. *Advances in Engineering Software*, 45(1):272–280, 2012.

[5] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, 3(2):149–156, 1991.

[6] M. Asghari and S. Nezhadali. Fuzzy multi-objective parallel machines scheduling problem through new solution method.

[7] K. R. Baker and D. Trietsch. *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[8] S. Balin. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. *Information Sciences*, 181(17):3551–3569, 2011.

[9] S. Balin. Non-identical parallel machine scheduling with fuzzy processing times using robust genetic algorithm and simulation. *Int J Innov Comput Inf Control*, 8(1):727–745, 2012.

[10] J. Cadenas and J. Verdegay. Modelos de optimización con datos imprecisos. *Univesidad de Murcia, Servicio de Publicaciones*, 1999.

[11] J. A. Carballido, I. Ponzoni, and D. Brignole. Un algoritmo genético basado en números difusos triangulares. In *VII Congreso Argentino de Ciencias de la Computación*, 2001.

[12] S. Chanas and A. Kasperski. Minimizing maximum lateness in a single machine scheduling problem with fuzzy processing times and fuzzy due dates. *Engineering Applications of Artificial Intelligence*, 14(3):377–386, 2001.

[13] S. Chanas and A. Kasperski. On two single machine scheduling problems with fuzzy processing times and fuzzy due dates. *European Journal of Operational Research*, 147(2):281–296, 2003.

[14] S. Chanas and A. Kasperski. Possible and necessary optimality of solutions in the single machine scheduling problem with fuzzy parameters. *Fuzzy Sets and Systems*, 142(3):359–371, 2004.

[15] C. Chekuri and S. Khanna. Approximation algorithms for minimizing average weighted completion time. 2004.

[16] G. Deepak, S. Sameer, and S. Bala. Specially structured two stage flow shop scheduling to minimize the rental cost. *International Journal of Emerging trends in Engineering and Development*, 1(2):206–215, 2012.

[17] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The annals of mathematical statistics*, pages 325–339, 1967.

[18] D. Dubois, H. Fargier, and H. Prade. Fuzzy constraints in job-shop scheduling. *Journal of intelligent Manufacturing*, 6(4):215–234, 1995.

[19] D. Dubois and H. Prade. Ranking fuzzy numbers in the setting of possibility theory. *Information sciences*, 30(3):183–224, 1983.

[20] D. Dubois and H. Prade. The mean value of a fuzzy number. *Fuzzy sets and systems*, 24(3):279–300, 1987.

[21] A. Duenas and D. Petrovic. Multi-objective genetic algorithm for single machine scheduling problem under fuzziness. *Fuzzy Optimization and Decision Making*, 7(1):87–104, 2008.

[22] H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, 3(2):225–251, 1963.

[23] P. Fortemps. Jobshop scheduling with imprecise durations: a fuzzy approach. *Fuzzy Systems, IEEE Transactions on*, 5(4):557–569, 1997.

[24] N. González, C. R. Vela, and I. González-Rodríguez. Comparative study of meta-heuristics for solving flow shop scheduling problem under fuzziness. In *Bio-inspired Modeling of Cognitive Tasks*, pages 548–557. Springer, 2007.

[25] I. R. Goodman. *Fuzzy sets as equivalence classes of random sets*, volume 327. Pergamon Press, Oxford, 1982.

[26] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[27] S. Gupta, Aggarwal. A fuzzy logic based approach to minimize the rental cost. *Pelagia Research Library*, pages 1986–1991, 2012.

[28] S. Gupta, Aggarwal. A fuzzy logic based approach to minimize the rental cost of machines for specially structured three stages flow shop scheduling. *Advances in Applied Science Research*, pages 1071–1076, 2012.

[29] S. Heilpern. Interval random sets and entropy. *Fuzzy sets and systems*, 35(2):213–217, 1990.

[30] S. Heilpern. The expected value of a fuzzy number. *Fuzzy sets and Systems*, 47(1):81–86, 1992.

[31] H. Ishii and M. Tada. Single machine scheduling problem with fuzzy precedence relation. *European Journal of Operational Research*, 87(2):284–288, 1995.

[32] T. Itoh and H. Ishii. One machine scheduling problem with fuzzy random due-dates. *Fuzzy Optimization and Decision Making*, 4(1):71–78, 2005.

[33] V. S. Jadhav and V. Bajaj. Single machine scheduling problem under fuzzy processing time and fuzzy due dates.

[34] M. JIMÉNEZ. Ranking fuzzy numbers through the comparison of its expected intervals. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 4(04):379–388, 1996.

[35] A. Kasperski. The fuzzy single machine scheduling problem-some general properties. In *EUSFLAT Conf.*, pages 577–581, 2003.

[36] G. Klir and T. Folger. *Fuzzy sets, uncertainty, and information*. Prentice Hall, 1988.

[37] G. Klir and B. Yuan. *Fuzzy sets and fuzzy logic*. Prentice Hall New Jersey, 1995.

[38] M. Kuroda and Z. Wang. Fuzzy job shop scheduling. *International Journal of Production Economics*, 44(1):45–51, 1996.

[39] E. L. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in operations research and management science*, 4:445–522, 1993.

[40] S. Lawrence. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). *Graduate School of Industrial Administration*, 1984.

[41] D. Lei. Fuzzy job shop scheduling problem with availability constraints. *Computers & Industrial Engineering*, 58(4):610–617, 2010.

[42] Z. Li and M. Ierapetritou. Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, 32(4):715–727, 2008.

[43] C. S. McCahon and E. S. Lee. Fuzzy job sequencing for a flow shop. *European Journal of Operational Research*, 62(3):294–301, 1992.

[44] M. S. Mehrabad and A. Pahlavani. A fuzzy multi-objective programming for scheduling of weighted jobs on a single machine. *The International Journal of Advanced Manufacturing Technology*, 45(1-2):122–139, 2009.

[45] K. Muthusamy, S. C. Sung, M. Vlach, and H. Ishii. Scheduling with fuzzy delays and fuzzy precedences. *Fuzzy Sets and Systems*, 134(3):387–395, 2003.

[46] S. Nasseri, E. Behmanesh, F. Taleshian, M. Abdolalipoor, and N. TaghiNezhad. Fully fuzzy linear programming with inequality constraints. *International Journal of Industrial Mathematics*, 5(4):309–316.

[47] E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91(1):160–175, 1996.

[48] J. J. Palacios, J. Puente, C. R. Vela, and I. González-Rodríguez. Benchmarks for fuzzy job shop problems. *Information Sciences*, 329:736–752, 2016.

[49] J. Peng and B. Liu. Parallel machine scheduling models with fuzzy processing times. *Information Sciences*, 166(1):49–66, 2004.

[50] S. Petrovic, C. Fayad, D. Petrovic, E. Burke, and G. Kendall. Fuzzy job shop scheduling with lot-sizing. *Annals of Operations Research*, 159(1):275–292, 2008.

[51] M. Pinedo. *Scheduling*. Springer, 2015.

[52] M. L. Pinedo. Scheduling: theory, algorithms, and systems. 2012.

[53] J. Razmi, M. Saffari, et al. A mathematical model for a flow shop scheduling problem with fuzzy processing times. *Journal of Optimization in Industrial Engineering*, pages 39–44, 2010.

[54] M. Sakawa and R. Kubota. Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. *European Journal of operational research*, 120(2):393–407, 2000.

[55] M. Sakawa and T. Mori. An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate. *Computers & Industrial Engineering*, 36(2):325–341, 1999.

[56] M. Schneider and A. Kandel. Properties of the fuzzy expected value and the fuzzy expected interval in fuzzy environment. *Fuzzy sets and systems*, 28(1):55–68, 1988.

[57] Y. Sotskov and F. Werner. A stability approach to sequencing and scheduling under uncertainty. *Sequencing and Scheduling with Inaccurate Data. Eds. YN Sotskov and F. Werner. Nova Science Publishers, Inc., New York, USA*, 2014.

[58] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993.

[59] R. Tavakkoli-Moghaddam, B. Javadi, F. Jolai, and A. Ghodratnama. The use of a fuzzy multi-objective linear programming for solving a multi-objective single-machine scheduling problem. *Applied soft computing*, 10(3):919–925, 2010.

[60] E. Trillas Ruiz, C. Alsina Catalï¿½, and J. Terricabras. *Introduccï¿½n a la lï¿½gica borrosa*. Ariel, 1995.

[61] H.-C. Wu. Solving the fuzzy earliness and tardiness in scheduling problems by using genetic algorithms. *Expert Systems with Applications*, 37(7):4860–4866, 2010.

[62] R. R. Yager. A note on probabilities of fuzzy events. *Information Sciences*, 18(2):113–129, 1979.

[63] L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.